PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 17/30 — A2

(11) International Publication Number: **WO 00/65486**

(43) International Publication Date: 2 November 2000 (02.11.00)

(21) International Application Number: PCT/US00/09203

(22) International Filing Date: 7 April 2000 (07.04.00)

(30) Priority Data:
60/128,466 9 April 1999 (09.04.99) US

(71) Applicant *(for all designated States except US)*: SANDPIPER SOFTWARE, INC. [US/US]; 1901 S. Bascom Avenue, Suite 700, Campbell, CA 95008 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: LAI, Eric [US/US]; Fremont, CA (US). KENDALL, Elisa, F. [US/US]; Santa Cruz, CA (US). WONG, James, S. [US/US]; Redwood City, CA (US). WONG, Benson, T. [US/US]; Belmont, MA (US).

(74) Agents: MILLIKEN, Darren, J. et al.; Blakely, Sokoloff, Taylor & Zafman LLP, 7th floor, 12400 Wilshire Boulevard, Los Angeles, CA 90025 (US).

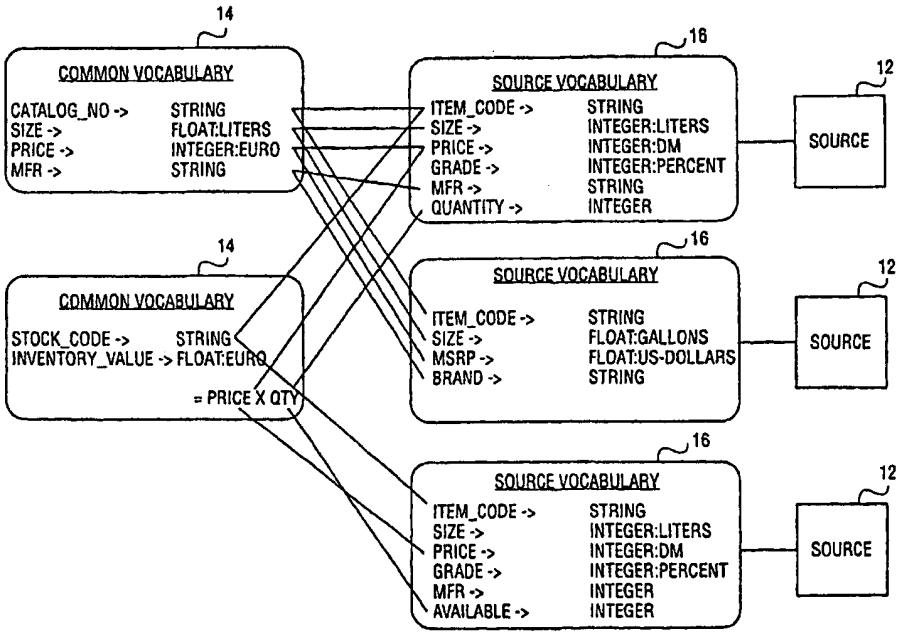
(81) Designated States: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: A METHOD OF MAPPING SEMANTIC CONTEXT TO ENABLE INTEROPERABILITY AMONG DISPARATE SOURCES

(57) Abstract

A distributed collection of applications and repositories with dissimilar syntactic and semantic characteristics may be accessed as though they are a single entity. Queries are performed transparently, without requiring the user to understand or be aware of the characteristics of any individual source. This is performed through the use of a common vocabulary and semantic mappings to the source repositories. A common vocabulary is used to provide the desired transparency. Mappings establish the correspondence between the common vocabulary and each of the source-specific vocabularies which are to be integrated. The common vocabulary may be tailored to the needs of an individual or group of users and/or applications and forms the basis for resolution of syntactic and semantic conflicts and ambiguities within a federation of information resources. The terms and constraints defined by the common vocabulary are then used to construct the query. The terms in the common vocabulary are mapped to each applicable sources vocabulary for query and retrieval purposes. By mapping the semantic characteristics of the various sources to the common vocabularly, the integrity of the individual source repositories, file systems, processes, and applications is maintained, while new applications and access methodologies can make use of the integrated resources.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

A METHOD OF MAPPING SEMANTIC CONTEXT TO ENABLE INTEROPERABILITY AMONG DISPARATE SOURCES

The present application claims the benefit of the filing date of U.S. provisional patent application serial no. 60/128,466 entitled "A METHOD OF MAPPING SEMANTIC CONTEXT TO ENABLE INTEROPERABILITY AMONG DISPARATE SOURCES", filed April 9, 1999.

STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH AND DEVELOPMENT

The invention was made with Government support under DAAL01-93-C-3380 awarded by the Defense Advanced Research Project Agency (DARPA). The Government has certain rights in the invention.

FIELD OF THE INVENTION

The invention relates to the field of data retrieval from databases (or other electronic sources), database or file management, and information processing technology. More specifically, it allows data to be accessed and/or exchanged between independently developed databases and applications, and it facilitates the integration of information, processes and applications.

BACKGROUND OF THE INVENTION

The structural and syntactic characteristics of an electronic data source may describe its data types, layout, composition, and some constraints on its access. The semantic properties of an application or repository define how that particular source should be interpreted. Semantic properties may reflect functional, behavioral, contextual, environmental, or content-specific characteristics of a process, application, or repository. An attribute name, for example, may have both syntactic and semantic properties. Other semantic (or context-related) characteristics may include units-of-measure, storage and precision for numeric data, definitions of terms, relationships among terms and values, rules for expression interpretation and evaluation, or localization characteristics. Structure and syntax may be explicitly defined through a database schema, data dictionary, or in source code. Semantic characteristics may be specified explicitly (*e.g.*, in an attribute name), implicitly (*e.g.*, units-of-measure, relationships among terms or objects, constraints on values), or not at all.

Many information systems, data warehouses, databases, and applications are developed independently to solve particular business problems. As business

requirements change and new functionality is required, the traditional methods for providing the requisite features have included:

- augmenting an existing system, or in some cases, new development that is consistent with existing formats;
- migrating source repositories to completely new environments or systems; and/or
- integrating multiple sources through the use of custom adapters that transform the data for each unique information producer, consumer, and repository. **Figure 1** is a block diagram showing a number of point-to-point integrations between two client applications 10, and a number of data sources 12. In each case, a custom adapter is provided between each client application 10 and each data source 12.

These tasks become increasingly difficult and more costly as the number and types of new and existing sources to be integrated increases. Even in cases where syntactic differences can be resolved through gateway or middleware technologies, for example, semantic conflicts and ambiguities remain. While many syntactic or structural details can be extracted from a database schema or through reverse engineering tools, semantic details that are implicitly specified or insufficiently documented are lost.

Additionally, individuals and information systems may use specific terminology to refer to objects and concepts. Cultural and organizational heritage, domain expertise, and educational and professional background all affect the language that an individual or group uses on a daily basis. The use of synonymous terms and terms whose definition is similar but not identical are common among geographically distributed organizations. Examples of terms whose usage reflects these issues are included in Table 1.

Table 1 Examples of Semantic Ambiguities - Terminology

Attribute	Possible Meanings	Issue
PC Card	(1) a printed circuit card, (2) any personal computer circuit card, (3) a PCMCIA peripheral card	multiple meanings within the same or similar domain(s)
Apple	(1) a type of fruit, (2) a computer company, (3) an early computer, (4) a record company	distinct, domain-specific meanings
User Interface	MMI; CHI; HCI; GUI; UI	synonymous terms
Satellite	Spacecraft	similar but not identical (a probe is considered a spacecraft but not a satellite)
CIA	(1) Central Intelligence Agency (2) Culinary Institute of America	two organizations sharing a common acronym
Lockheed Martin Corp.	LMT (its stock symbol)	domain specific symbology
ARPA	DARPA	an organization that has renamed itself more than once during its history
Province	state	geographically (or nationally) specific but similar concepts

In other cases, the interpretation of numerical data may be left to the user's implicit understanding of the data source. Some examples appear in Table 2.

Table 2 Examples of Semantic Ambiguities – Units and Normalization

Attribute	Units
Price	US Dollars; Canadian Dollars; Euro
Weight	ounces; pounds; kilogrammes
Memory Size	Kilobytes; Megabytes; Megabits
Data Transmission Rate	Baud (*i.e.*, symbols per second) bits per second
Data presented in financial reports	stated in Dollars; normalized to thousands of Dollars; normalized to millions of dollars

As resources are increasingly accessible across organizational and national boundaries, users of those resources may find it difficult to determine the correct units

or semantic context appropriate for a particular data element. For example, a search of two independently-developed databases for pricing information such as price greater then $100.00 with results merged for comparison purposes, may yield results that are inconsistent or inaccurate, or that lead to incorrect interpretation. In cases where the units are labeled, there is no mechanism available to the user to easily compare the results without manual translation.

Migrating data from one environment to another is typically expensive, time-consuming, disruptive, and error-prone, frequently resulting in loss of data or functionality. Up to 80 percent of the cost of development of a data warehouse is expended in data extraction, cleaning and loading. The migration process includes modeling existing processes and information sources, designing new processes and information models, designing the new schema, developing data conversion tools, and creating, converting, loading, testing, and validating the new repository or application. It often requires service outages for both anticipated tasks and unanticipated problems. Converted repositories are frequently incompatible with existing sources; database synchronization issues often arise during transition intervals, where both the original and new systems operate in parallel. Conversion processes may require six to eighteen months to complete, though it is not uncommon for enterprise-scale projects to take more than two years.

Various methods are employed by commercial software integration tools to resolve syntactic and structural differences among information sources, including database gateways, message-oriented middleware, and application-specific adapters. These methods fail to resolve semantic conflicts, inconsistencies, and ambiguities that naturally occur among independently developed information systems, however. Semantic integration issues are likely to arise when resources use differing naming conventions, where there may be distinctions in terminology and taxonomies, where the same terms may be used inconsistently or have differing properties (such as different units of measure, enumerated values, or constraints). These mismatches are far more difficult to resolve than syntactic and structural incompatibilities, and necessitate costly, custom solutions at best. The approach promoted by most systems integrators today is to define a single, monolithic data model for use throughout an organization, which, while effectively eliminating semantic conflicts, either forces the organization to migrate all of its information to the new model or prohibits integration with autonomously developed sources. This approach is not always feasible due to cost, resource constraints, or business requirements to maintain legacy systems. Semantic characteristics are often implicitly represented in the data and are seldom well documented. Failing to address semantic or context- or content-specific integration issues may produce unsatisfactory results and may have a negative impact on systems integrity.

Natural language processing methods for the purposes of understanding and retrieving information from various static, dynamic, and interactive sources exist. These methods represent attempts to generalize (or refine) a set of query terms, to expand a set of query terms, or to construct computable queries from expressions used in normal human discourse. While these approaches support customization from the user's perspective, they do not support transparent querying or integration of disparate sources without requiring the user to have knowledge of the source specific terminology.

SUMMARY OF THE INVENTION

According to the present invention, there is provided a method of querying a plurality of data sources. A common query is received, the common query being constructed utilizing a common vocabulary that is mapped to a plurality of source vocabularies associated with respective data sources of the plurality of data sources. A plurality of source queries are generated to the plurality of data sources, each source query being derived from the common query and being constructed utilizing a respective source of vocabulary.

Other features of the present invention will be apparent from the accompanying drawings and from the detailed description that follows.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example and not limitation in the figures of the accompanying drawings, in which like references indicate similar elements and in which:

Figure 1 is a block diagram illustrating a prior art point-to-point integration scheme between a plurality of data consumers and a plurality of data sources.

Figure 2 is a block diagram illustrating the use of a common vocabulary to provide a two-level integration between a number of data consumers and a number of data sources, according to one embodiment of the present invention.

Figure 3 is a block diagram illustrating mappings between two common vocabularies and a plurality of source vocabularies, according to one embodiment of the present invention.

Figure 4A is a diagrammatic representation of an exemplary client-server (or two-tiered) environment within which the present invention may be deployed.

Figure 4B is a diagrammatic representation of an exemplary three-tier environment within which the present invention may be deployed.

Figure 5 is a block diagram illustrating architectural details of an exemplary data server constructed to implement one embodiment of the present invention.

Figure 6 is a flow chart illustrating the primary operations of an exemplary method of querying a plurality of data sources, and receiving a response to such a query, according to one embodiment of the present invention.

Figure 7 is a flow chart illustrating a method of generating a source query expression for an applicable source, the source query expression being derived from a common query expression, according to one embodiment of the present invention.

Figure 8 is a flow chart illustrating an exemplary method, according to one embodiment of the present invention, of performing attribute mapping function.

Figure 9 is a flow chart illustrating an exemplary method, according to one embodiment of the present invention, of performing a semantic mapping for algorithmic conversion of attribute values.

Figure 10 is a flow chart illustrating an exemplary method, according to one embodiment of the present invention, of performing semantic mapping for lookup-based mapping of attribute values.

Figure 11 is a diagrammatic representation of an exemplary attribute mapping table, according to one embodiment of the present invention.

Figure 12 is a block diagram illustrating an exemplary machine, in the form of a computer system, that may execute a sequence of instructions embodying the present invention and that includes a machine-readable medium for storing such a sequence of instructions.

DETAILED DESCRIPTION

Overview

A method and apparatus for querying a plurality of data sources are described. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will

be evident, however, to one skilled in the art that the present invention may be practiced without these specific details. The invention, in one embodiment, enables a distributed collection of applications and repositories with dissimilar syntactic and semantic characteristics to be accessed as though they are a single entity. Queries are performed transparently, without requiring the user to understand the characteristics of any individual source, through the use of a common vocabulary and semantic mappings to the source repositories. A common vocabulary is used to provide the desired transparency. Mappings establish the correspondence between the common vocabulary and each of the source-specific vocabularies which are to be integrated. The common vocabulary may be tailored to the needs of an individual or group of users and/or applications and forms the basis for resolution of syntactic and semantic conflicts and ambiguities within a federation of information resources. The terms and constraints defined by the common vocabulary are then used to construct the query. The terms in the common vocabulary are mapped to each applicable source vocabulary for query and retrieval purposes. By mapping the semantic characteristics of the various sources to the common vocabulary, the integrity of the individual source repositories, file systems, processes, and applications is maintained, while new applications and access methodologies can make use of the integrated resources.

A method of using semantic vocabularies and mappings is described. This approach allows data systems to be integrated or accessed transparently, from a single point of entry, despite differences in the semantic characteristics of the applications or repositories.

In one exemplary embodiment, the requisite vocabularies and semantic mappings may be prepared and information sources may be integrated using the following operations:

(1) Each data source to be integrated is modeled to create a source vocabulary. The vocabulary is essentially a structural, syntactic and semantic schema for the source.

(2) A common vocabulary, containing any combination of terms and attributes from the source(s), and the relationship and constraints that formally define those terms, is created. The characteristics of the common vocabulary need not be the same [in structure, syntax, or semantics] as any of the individual source vocabularies. It should be noted that the terminology, formats and semantics of the common vocabulary may be determined by one or more domain-specific standards, organizational requirements, or the user's preference.

(3) The common vocabulary is mapped to each of the source vocabularies.

(4) The mappings are converted to a form that is usable by the run-time engine. This process will take the mapping between the names in the source vocabulary and variables in the common vocabulary (created in operation 3) and generate the contents of the mapping tables. The details for each vocabulary item are retrieved from the source models and the common models created in operations 1 and 2. The structural, syntactic and semantic correlations between a particular pair of vocabularies will be encoded. In the case of mappings performed by algorithms (*e.g.*, units-of-measure), it will be necessary determine if a mapping exists between the two forms. If not, this fact will be flagged for the knowledge engineer. In this case it is the responsibility of the knowledge engineer to create a mapping and update the list of mappings available to the knowledge base used by the generation process.

Allowing the vocabulary to be tailored enables the user to access and evaluate the information using familiar terminology and conventions. The use of a common vocabulary addresses the need for each unique client application to be integrated with each unique source. Furthermore, it allows the set of sources sharing this vocabulary to be queried transparently through a single query which need not be reformulated for each target repository. The user (and the client application) need not understand the specific details required to access each source. This approach allows sources to be integrated incrementally. Once a common vocabulary has been established, sources may be integrated at a pace determined by business needs. The common vocabulary can evolve over time as additional sources are integrated or new business requirements arise. Multiple common vocabularies, and thus multiple federated collections of applications and repositories, may be integrated using the same approach.

An exemplary use of the common vocabulary may be the semantic integration of legacy sources. A common vocabulary may also serve as a starting point from which new source systems are developed.

At run-time, the following operations are performed:

(1) A pre-existing common vocabulary is selected based on the problem domain, relevant business processes and requirements, the available source repositories and applications, and user preferences, as appropriate. The user (or application) may also select a subset of the available sources for searching through the use of a directory services mechanism.

(2) A query is formulated using the terms and constraints of the common vocabulary.

(3) Identical queries are distributed, in parallel, to each applicable source repository.

(4) Upon receipt, each server maps the parameters specified in the request from the common vocabulary to the source-specific vocabulary. This process may include term renaming, unit conversion, translation, or mapping attributes and values to those used by the source.

(5) The source application, database management system, file system, or other information processing system acts on the request and returns the result set (if any).

(6) The terms of the result set are mapped from the local vocabulary to the common vocabulary, and

(7) the result set is returned to the client for further processing.

As a mapping exists from each individual source vocabulary to the common vocabulary, queries may be performed transparently across the set of integrated sources. This arrangement allows modular, plug-in, incremental integration of new sources into an existing environment. By creating a common vocabulary for the existing system, new sources may be mapped to the common vocabulary without requiring modification of those sources. No changes are required to existing applications to allow a new source to be utilized. **Figure 1** illustrates the complexity of integration using the traditional, custom approach. The present invention is advantageous in that it may substantially reduce the number of point-to-point integrations needed among applications and repositories that comprise the user's environment. **Figure 3** illustrates the use of a common vocabulary 14 to resolve attribute names, data types and units between a number of sources 12 and a number of client (user) applications 10. Specifically, **Figure 3** illustrates the use of two independent, common vocabularies 14 which map combinations of attributes from three source vocabularies 16, each source vocabulary 12 being associated with a respective source 12. In one embodiment, each of the respective sources may display dissimilar structural syntax, semantics or access protocols from other source vocabularies 16.

The mapping component may be relocated to an application server servicing a cluster of sources 12 with minimal loss of flexibility. Whether vocabulary mapping functionality is performed by the information source server or a middle-tier application server, client applications (and therefore users) 10 are shielded from the specifics of the individual source vocabularies 16.

Exemplary Embodiment

Figure 4A illustrates an exemplary client-server environment 20 in which the present invention may be used. The environment 20 includes one or more client applications 10 which may utilize individual or shared vocabularies, and multiple data sources in the exemplary form of data servers 22 which may utilize individual or shared

vocabularies. Server vocabularies of the data servers 22 may overlap with the client vocabularies of the client applications 10 although this need not be the case. The data servers 22 may have dissimilar structure and syntax, dissimilar semantics and terminology, and dissimilar access protocols from the client applications 10. In one embodiment, at least one data server 22 supports one client vocabulary, and the client applications 10 and servers 22 are connected by a network 24 or other communication medium. The servers 22 (or a subset thereof) may share at least one common vocabulary 14. This common vocabulary 14 is utilized by at least one client application. There may be multiple common vocabularies. The term "common" is with respect to some set of data servers or sources.

Requests 26 are sent from the client application 10 to the server 22 and responses 28 from the servers 22 to the client application 10. The client application 10 generates requests 26 and receives responses 28 in the user's terminology and semantics (*i.e.*, the common vocabulary). Each data server 22 may receive requests 26 and generate responses 28 in the user terminology and semantics. However, the internal repositories utilizes the source-specific local terminology and semantics. It should be noted that a user terminology and semantics set is not arbitrary, rather it is a set of corresponding terms, concepts and preferences which have been tailored to one or more users. Each data server 22 sharing the common vocabulary 14 (or a user selected subset of the data servers 22) receives an identical request 26.

The exemplary architecture, illustrated in **Figure 4A**, is client-server. The present invention is equally applicable to a three-tiered architecture such as that shown in **Figure 4B**, where the mapping functionality is performed on (one or more) intermediate application servers 30. The application server 30 receives client requests, maps the requests and queries the appropriate data sources (e.g., the servers 22), then remaps and returns the results to the client. An application server 30 is also capable of manipulating the results, if necessary, to sort, merge or filter the results as well as performing report generation and analysis functions. The application server 30 understands the mappings between each common vocabulary and each source vocabulary accessible in the federation. In the client-server case illustrated in **Figure 4A**, each server 22 is required to understand only those mappings between supported common vocabularies and its local vocabulary. The application server 30 also supports all of the access protocols needed to access any of the data servers 22. By using a server-based process, the client-server architecture takes advantage of a common access protocol. Since the application server 30 intermediates each transaction, it may require relatively powerful hardware and may potentially become a bottleneck (depending on the number of users or the tasks performed on behalf of users) or single-point-of-failure. An application server 30 does, however, allow the mapping information and business applications to be centralized which may reduce the maintenance to clients and back-

end servers. Unlike the client-server architecture, the three-tier approach does not require a server-resident mapping process be added to each data server 22. Using a client-server or three-tiered architecture, the concept of integrating dissimilar information sources using a common vocabulary remains unchanged. It will be appreciated that, in the three-tier architecture illustrated in **Figure 4B**, certain of these components will be deployed on the application server 30.

Figure 5 shows the primary components of an exemplary data server 22 as displayed within the two-tier (client-server) architecture shown in **Figure 4A**. The illustrated primary components of the exemplary data server 22 include a communication interface 32, a mapper 34 and a source interface 36 (*e.g.*, a user API, a DBMS) to the information repositories 38. The communication interface 32 is arbitrary and supports whatever communication methodologies (*e.g.*, TCP/IP, CORBA, ATM, Ethernet) is required to communicate over the network 24. The communication interface 32 also provides services to format/extract information to/from messages. As previously discussed, these messages utilize the users' terminology and semantics set. As illustrated in **Figure 5**, the mapper 34 transforms the users' terminology and semantics embodied in a request 26 into the local terminology and semantics (and vice versa). Once translated into the source-specific local terminology, a query is made to the appropriate data source 38 (e.g., database, file or other data management service). In order for the mapper 34 to perform the necessary translations, a number of mapping tables 40 are made available to it.

Since the client application 10 has knowledge of only the common vocabulary 14, the capability of bi-directional mapping is provided. The top half of **Figure 5** shows the mapping requests 26 from common to source vocabulary, while the lower half shows the mapping responses 28 from source to common vocabulary.

While not required for the basic functionality, a variable length string may be used to transfer the requests and responses between the client application 10 and the data server 22. In the data server 22, lexical tokens, representing attribute names, operators and values may be extracted by a lexical analyzer 42, after which the mapping appropriate to the token (*e.g.*, name or value) is performed. A syntax analyzer 44 determines whether a token is an attribute name, an operator, a value or something else. An alternative is to remember what the last token was (assuming the use of <attribute_name, operator, attribute_value> triples). The mapping functions may be built as actions to the lexical analyzer 42 and generation of the query expression may be performed by actions in the syntax analyzer 44. Relational expressions may be chained together in compound expressions as shown in **Figure 7**. The input and output grammars are chosen by the user according to the application. The lexical analyzer 42 and syntax analyzer 44 skeletons may be generated with commercially available tools

used for that purpose (e.g., AT&T lex and yacc). These skeletons may then augmented with the appropriate set of actions.

Figure 6 is a flow chart illustrating an overview of an exemplary method 50 of performing a mapping of a request 26, expressed in a common vocabulary, to a source vocabulary 16. The method 50 commences with lexical extraction operation 52 by the lexical analyzer 42, followed by a semantic mapping operation 54 performed by the mapper 34 and a syntax analysis operation 56 performed by the syntax analyzer 44.

Two types of semantic mapping are performed as part of the semantic mapping operation 54 in order to provide the desired transparency, namely attribute name mapping 60 and attribute value mapping 62. Attribute name mapping 60, further illustrated in **Figure 8**, resolves naming differences between the attributes in the vocabularies. Attribute value mapping 62 resolves differences between the contents of the attributes. Attribute value mapping 62 may be performed in one of two exemplary methods: algorithmically, as illustrated in **Figure 9**, or by lookup, as illustrated in **Figure 10**.

Figure 7 is a flow chart illustrating further details of a method 66, according to one embodiment of the present invention, of expression generation. It should be noted that **Figure 7** does not illustrate lexical, syntactic or semantical analysis performed through compiler functionality, nor symbol table management or generation provided by compiler technology or techniques.

The method 66 illustrates the chaining together of relational expressions into a compound expression. The method 66 commences with a mapping operation 54, followed by a determination at decision block 68 as to whether a logical operation has been detected. If so, the method 66 continues with the logical expression formation by proceeding via block 70 to perform a further mapping operation 54. On the other hand, should no logical operation be detected at decision box 68, a compound expression, comprising a chained series of relation expressions, is submitted at block 72.

As stated above, two types of semantic mapping are performed as part of the semantic mapping operation. **Figure 8** is a flow chart illustrating a method 71, according to an exemplary embodiment of the present invention, of performing an attribute name mapping 60.

The method 71 commences at block 72 with a lookup attribute name operation. Specifically, a global name 74 is utilized to perform a lookup with respect to an attribute name table 78, and to return a local name 76.

As shown in **Figure 11**, there should be at least one entry within the attribute name table 78 for each attribute which stores the mapped global name 74, the mapped data type and a string which specifies any required algorithmic conversion(s). A key consisting of the vocabulary name, the class name and the attribute (or local) name 76 concatenated together is used as a key to a database lookup. This key structure allows

multiple vocabularies to be stored in a single database or table. It also allows the same attribute name 76 to exist in different classes as well as for the same class name to exist in different vocabularies. Once the mapping information is retrieved, the attribute (local) name 76 found in the attribute name table 78 is written to an output string followed by the detected relational operator.

Returning to **Figure 8**, at decision box 80, a determination is made as to whether the local name is valid for a relevant source vocabulary 16 (i.e., does the local name correspond to an actual name within the source vocabulary 16). If not, an error is generated at block 82. Alternatively, at block 84, the local name 76 is inserted into the query expression.

At block 86, the attribute data type (e.g., floating point number, string or integer) is retrieved. At block 88, formatting of the expression commences.

The lookup attribute value mapping 62 also includes two components, namely (1) a semantic mapping for algorithmic conversion component and (2) a semantic mapping for lookup-based mapping of attribute values component.

Figure 9 is a flow chart illustrating an exemplary method 90, according to one embodiment of the present invention, of performing a semantic mapping of algorithmic conversions on attribute values.

At block 92, upon the detection of a value token (or operand), the lexical type for the relevant value token is determined.

In a first instance, if the value associated with the value token is typed as a floating point number, an algorithmic conversion, if indicated by the mapping structure, is performed at block 94. Similarly, if the value associated with the value token is typed as an integer, an algorithmic conversion, if indicated by the mapping structure, is performed at block 96. To perform the algorithmic conversions at blocks 94 and 96, the mapper 34 has access to a library of supported conversions integrated into it.

Examples of algorithmic conversions that may be performed at blocks 94 and 96 include unit-of-measure conversions (e.g., centimeters to inches) or any other conversions of values that may be algorithmically performed, as opposed to requiring a lookup operation.

Figure 10 is a flow chart illustrating an exemplary method 100, according to one exemplary embodiment of the present invention, of performing a semantic mapping for lookup-based mapping of attribute values.

As noted, if a value is typed as a floating point number at block 92 in **Figure 9**, is subject to an algorithm conversion at block 94. **Figure 10** reveals that the value receives no further value mapping, and is accordingly inserted into an output string and a next token value is located.

On the other hand, for string or integer typed values, additional lookups may be performed. A key (not shown) consisting of a vocabulary name, a class name, an attribute name and the value are concatenated together and utilized as a key to a database lookup in the name table 78 or a value table 79.

Dealing first with a string typed value, two attempts may be made to map such a string value. First, at block 102, a lookup is made in the attribute name table 78 to determine whether or not the value is in fact another attribute (local) name. If the lookup at block 102 succeeds, as determined at decision block 104, the local name 76 substitutes the global name 74, the result is placed in an output string, and the next value token is sorted.

On the other hand, should it be determined at decision block 104 that the lookup at block 102 has failed, the global name 74 is retained at block 108 and the key is utilized to perform a second lookup at block 110 in an attribute value table 79 to determine if the key is to an enumerated value.

If the lookup operation at block 110 succeeds, as determined at decision block 112, then a the local value retrieved from the value table 79 is substituted for the global value, written to an output string, whereafter the next value token is sought.

On the other hand, should it be determined at decision block 112 that the lookup operation at block 110 has failed, then the original, global value is retained at block 116, and written into an output string. Thereafter, the next value token is sought.

In the case where the value token is determined at block 92 to comprise an integer type value, following any required algorithmic conversions at block 96, the actions described above with reference to blocks 110 - 116 are then performed with respect to such an integer type value.

It should be noted that the two stage mapping of string values is only needed for applications involving query generation (since the value of two attributes may be compared with one another). In an alternative embodiment, only a single lookup may be used. Specifically, if a value token representing a logical operator is located, the logical operator will be added to an output string, and the process repeated. When the end of a valid expression is detected, the output string is used to query the data source. The results returned from the data source are then mapped back into the common vocabulary 14. The reverse mapping process is substantially as described above, with two exceptions. First, the mapping tables (e.g., the name table 78 and the value table 79) are from the source (or local) vocabulary 16 to the common vocabulary 14. Secondly, the input and output are <attribute_name, attribute_value> tuples.

Figure 11 shows the structure of exemplary mapping tables 64 within the context of an object-oriented database. The same structure may be implemented for the attribute name and the attribute value tables 78 and 79. In the attribute name table 78, all three fields are needed. For the attribute value table 79, only the field for the

mapped value field is needed. A mapped name may furthermore be an attribute name or an attribute value. The tables 64 may be implemented through the object-oriented database using a hashed dictionary structure 130. Another data structure may be used so long as it has a look-up or search capability. The data structures, itself, need not be persistent, if it can be loaded from a persistent source (*e.g.*, a file) when required. A database does, however, provide built-in mechanisms to more easily manage, store, retrieve and update the data. It is clear that the choice of data structures may affect the performance of the mapping functions. The hashed dictionary structure 130 generally provides fast lookup performance relative to other commonly used methods if an effective hash function is selected. Since these tables 64 are of the same format and the key string allows sufficient uniqueness, the two tables may be merged into one. This approach, however, may not significantly reduce the number of lookups, the aggregate table size, or the maintenance requirements but may result in some loss of understandability.

As noted above, in order to perform the required algorithmic conversions, the mapper 34 has a library of supported conversions integrated into it. One embodiment of the present invention links a library (static or dynamic) of conversion algorithms into the mapper 34. The appropriate conversion function is selected based on a code in the conversion specifier string, as shown in **Figure 11.** These may be packaged as remote procedures or distributed functions such as the Object Management Group's Common Object Request Broker Architecture (CORBA). This integration allows new functions to be easily added.

Figure 12 is a block diagram illustrating a machine, in the exemplary form of a computer system 250, within which a set of instructions for causing the computer system 250 to perform any one of the methodologies discussed above may be executed. The computer system 250 includes a processor 252, a main memory 254, and a static memory 255, which communicate with each other via a bus 256. The computer system 250 further includes a video display unit 258 (e.g., a liquid crystal display (LCD) or a cathode ray tube (CTR)). The computer system 250 further includes an alpha-numeric input device 260 (e.g., a keyboard), a cursor control device 262 (e.g., a mouse), a storage medium in the exemplary form of a disk drive unit 264, a signal generation device 266 (e.g., a speaker) and a network interface device 268.

The disk drive unit 264 includes a machine-readable medium 265 on which is stored a set of instructions (i.e., software 270) embodying any one, or all, of the methodologies described above. The software 270 is also shown to reside, completely or at least partially, within the main memory 254 and/or within the processor 252. The software 270 may furthermore be transmitted or received via the network interface device 268. For the purposes of the present specification, the term "machine-readable medium" shall be taken to include any medium that is capable of storing and encoding

a sequence of instructions for execution by the machine, and that causes the machine to perform any one of the methodologies of the present invention. The term "machine-readable medium" shall accordingly be taken to include, but not limited to, solid-state memories, optical and magnetic disks, and carrier wave signals.

The present application is useful for developing and integrating applications and repositories wherein the data is accessed using terminology, methodologies, or other mechanisms that may differ from the source implementation, format, or context. It may be used to resolve both semantic and syntactic differences among applications and repositories. Although databases are used as examples throughout this disclosure, the approach is independent of the source implementation or underlying technology. This invention is also applicable to information brokering and mediation.

The present invention is also advantageous in that data migration is not required to customize the vocabulary or transparently to query multiple data sources as a single set. This alone results in a significant reduction of cost, operational disruption and risk. As the integrity of the various source repositories and applications is preserved, the operation of existing systems remains uninterrupted while new applications can access and make use of the information maintained by those systems.

While an exemplary embodiment of the present invention has been described above in the context of a query that is targeted to a set of structurally, syntactically or semantically diverse set of information sources, it will be appreciated that the invention extends to the integration of information sources through the use of a common vocabulary to resolve structural, syntactic or semantic differences between subsources and the customization and resolving of an interface to a set of structurally, syntactically or semantically diverse set of data sources utilizing terminology particular to a user's preference or task area.

Thus, a method and apparatus for querying a plurality of data sources have been described. Although the present invention has been described with reference to specific exemplary embodiments, it will be evident that various modifications and changes may be made to these embodiments without departing from the broader spirit and scope of the invention. Accordingly, the specification and drawings are to be regarded in an illustrative rather than a restrictive sense.

CLAIMS

What is claimed is:

1. A method of querying a plurality of data sources, the method including:

receiving a common query constructed utilizing a first common vocabulary, the first common vocabulary being mapped to a plurality of source vocabularies associated with respective data sources of the plurality of data source; and

generating a plurality of source queries to the plurality of data sources, each source query being derived from the common query and being constructed utilizing a respective source vocabulary.

2. The method of claim 1 wherein the common query is constructed utilizing a first term of the first common vocabulary, the first term of the first common vocabulary being mapped by at least one mapping structure to corresponding first terms of the plurality of source vocabularies associated with the respective data sources.

3. The method of claim 2 wherein the generating of the plurality of source queries includes identifying the corresponding first terms of the plurality of source vocabularies utilizing the first term of the first common vocabulary and the at least one mapping structure, and including the first terms of the plurality of source vocabularies within the respective source queries directed to the plurality of data sources.

4. The method of claim 1 wherein the first common vocabulary is semantically mapped to the plurality of source vocabularies so as to address semantic differences across the plurality of source vocabularies.

5. The method of claim 1 wherein the common vocabulary is syntactically mapped to the plurality of source vocabularies so as to address syntactic differences across the plurality of source vocabularies.

6. The method of claim 1 wherein the common vocabulary is structurally mapped to the plurality of source vocabularies so as to address structural differences across the plurality of source vocabularies.

7. The method of claim 1 wherein the generating of the plurality of source queries comprises performing a name translation for the first term between a common name and a source name.

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 17/30 | A2

(11) International Publication Number: WO 00/65486

(43) International Publication Date: 2 November 2000 (02.11.00)

(21) International Application Number: PCT/US00/09203

(22) International Filing Date: 7 April 2000 (07.04.00)

(30) Priority Data:
60/128,466 9 April 1999 (09.04.99) US

(71) Applicant *(for all designated States except US)*: SANDPIPER SOFTWARE, INC. [US/US]; 1901 S. Bascom Avenue, Suite 700, Campbell, CA 95008 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: LAI, Eric [US/US]; Fremont, CA (US). KENDALL, Elisa, F. [US/US]; Santa Cruz, CA (US). WONG, James, S. [US/US]; Redwood City, CA (US). WONG, Benson, T. [US/US]; Belmont, MA (US).

(74) Agents: MILLIKEN, Darren, J. et al.; Blakely, Sokoloff, Taylor & Zafman LLP, 7th floor, 12400 Wilshire Boulevard, Los Angeles, CA 90025 (US).

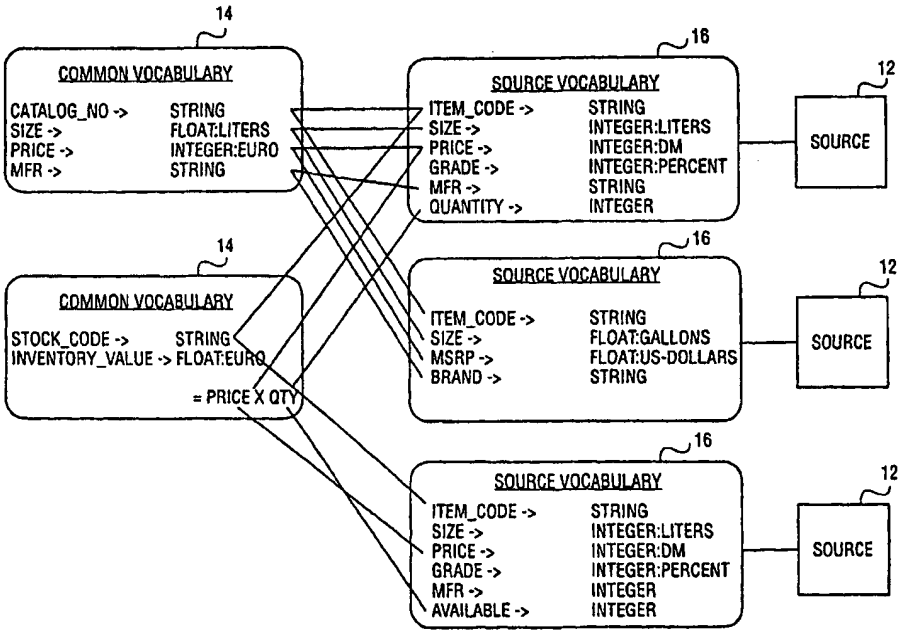
(81) Designated States: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: A METHOD OF MAPPING SEMANTIC CONTEXT TO ENABLE INTEROPERABILITY AMONG DISPARATE SOURCES

(57) Abstract

A distributed collection of applications and repositories with dissimilar syntactic and semantic characteristics may be accessed as though they are a single entity. Queries are performed transparently, without requiring the user to understand or be aware of the characteristics of any individual source. This is performed through the use of a common vocabulary and semantic mappings to the source repositories. A common vocabulary is used to provide the desired transparency. Mappings establish the correspondence between the common vocabulary and each of the source-specific vocabularies which are to be integrated. The common vocabulary may be tailored to the needs of an individual or group of users and/or applications and forms the basis for resolution of syntactic and semantic conflicts and ambiguities within a federation of information resources. The terms and constraints defined by the common vocabulary are then used to construct the query. The terms in the common vocabulary are mapped to each applicable sources vocabulary for query and retrieval purposes. By mapping the semantic characteristics of the various sources to the common vocabularly, the integrity of the individual source repositories, file systems, processes, and applications is maintained, while new applications and access methodologies can make use of the integrated resources.



A METHOD OF MAPPING SEMANTIC CONTEXT TO ENABLE INTEROPERABILITY AMONG DISPARATE SOURCES

The present application claims the benefit of the filing date of U.S. provisional patent application serial no. 60/128,466 entitled "A METHOD OF MAPPING SEMANTIC CONTEXT TO ENABLE INTEROPERABILITY AMONG DISPARATE SOURCES", filed April 9, 1999.

STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH AND DEVELOPMENT

The invention was made with Government support under DAAL01-93-C-3380 awarded by the Defense Advanced Research Project Agency (DARPA). The Government has certain rights in the invention.

FIELD OF THE INVENTION

The invention relates to the field of data retrieval from databases (or other electronic sources), database or file management, and information processing technology. More specifically, it allows data to be accessed and/or exchanged between independently developed databases and applications, and it facilitates the integration of information, processes and applications.

BACKGROUND OF THE INVENTION

The structural and syntactic characteristics of an electronic data source may describe its data types, layout, composition, and some constraints on its access. The semantic properties of an application or repository define how that particular source should be interpreted. Semantic properties may reflect functional, behavioral, contextual, environmental, or content-specific characteristics of a process, application, or repository. An attribute name, for example, may have both syntactic and semantic properties. Other semantic (or context-related) characteristics may include units-of-measure, storage and precision for numeric data, definitions of terms, relationships among terms and values, rules for expression interpretation and evaluation, or localization characteristics. Structure and syntax may be explicitly defined through a database schema, data dictionary, or in source code. Semantic characteristics may be specified explicitly (*e.g.*, in an attribute name), implicitly (*e.g.*, units-of-measure, relationships among terms or objects, constraints on values), or not at all.

Many information systems, data warehouses, databases, and applications are developed independently to solve particular business problems. As business

requirements change and new functionality is required, the traditional methods for providing the requisite features have included:

- augmenting an existing system, or in some cases, new development that is consistent with existing formats;
- migrating source repositories to completely new environments or systems; and/or
- integrating multiple sources through the use of custom adapters that transform the data for each unique information producer, consumer, and repository. **Figure 1** is a block diagram showing a number of point-to-point integrations between two client applications 10, and a number of data sources 12. In each case, a custom adapter is provided between each client application 10 and each data source 12.

These tasks become increasingly difficult and more costly as the number and types of new and existing sources to be integrated increases. Even in cases where syntactic differences can be resolved through gateway or middleware technologies, for example, semantic conflicts and ambiguities remain. While many syntactic or structural details can be extracted from a database schema or through reverse engineering tools, semantic details that are implicitly specified or insufficiently documented are lost.

Additionally, individuals and information systems may use specific terminology to refer to objects and concepts. Cultural and organizational heritage, domain expertise, and educational and professional background all affect the language that an individual or group uses on a daily basis. The use of synonymous terms and terms whose definition is similar but not identical are common among geographically distributed organizations. Examples of terms whose usage reflects these issues are included in Table 1.

Table 1 Examples of Semantic Ambiguities - Terminology

Attribute	Possible Meanings	Issue
PC Card	(1) a printed circuit card, (2) any personal computer circuit card, (3) a PCMCIA peripheral card	multiple meanings within the same or similar domain(s)
Apple	(1) a type of fruit, (2) a computer company, (3) an early computer, (4) a record company	distinct, domain-specific meanings
User Interface	MMI; CHI; HCI; GUI; UI	synonymous terms
Satellite	Spacecraft	similar but not identical (a probe is considered a spacecraft but not a satellite)
CIA	(1) Central Intelligence Agency (2) Culinary Institute of America	two organizations sharing a common acronym
Lockheed Martin Corp.	LMT (its stock symbol)	domain specific symbology
ARPA	DARPA	an organization that has renamed itself more than once during its history
Province	state	geographically (or nationally) specific but similar concepts

In other cases, the interpretation of numerical data may be left to the user's implicit understanding of the data source. Some examples appear in Table 2.

Table 2 Examples of Semantic Ambiguities – Units and Normalization

Attribute	Units
Price	US Dollars; Canadian Dollars; Euro
Weight	ounces; pounds; kilogrammes
Memory Size	Kilobytes; Megabytes; Megabits
Data Transmission Rate	Baud (*i.e.*, symbols per second) bits per second
Data presented in financial reports	stated in Dollars; normalized to thousands of Dollars; normalized to millions of dollars

As resources are increasingly accessible across organizational and national boundaries, users of those resources may find it difficult to determine the correct units

or semantic context appropriate for a particular data element. For example, a search of two independently-developed databases for pricing information such as price greater then $100.00 with results merged for comparison purposes, may yield results that are inconsistent or inaccurate, or that lead to incorrect interpretation. In cases where the units are labeled, there is no mechanism available to the user to easily compare the results without manual translation.

Migrating data from one environment to another is typically expensive, time-consuming, disruptive, and error-prone, frequently resulting in loss of data or functionality. Up to 80 percent of the cost of development of a data warehouse is expended in data extraction, cleaning and loading. The migration process includes modeling existing processes and information sources, designing new processes and information models, designing the new schema, developing data conversion tools, and creating, converting, loading, testing, and validating the new repository or application. It often requires service outages for both anticipated tasks and unanticipated problems. Converted repositories are frequently incompatible with existing sources; database synchronization issues often arise during transition intervals, where both the original and new systems operate in parallel. Conversion processes may require six to eighteen months to complete, though it is not uncommon for enterprise-scale projects to take more than two years.

Various methods are employed by commercial software integration tools to resolve syntactic and structural differences among information sources, including database gateways, message-oriented middleware, and application-specific adapters. These methods fail to resolve semantic conflicts, inconsistencies, and ambiguities that naturally occur among independently developed information systems, however. Semantic integration issues are likely to arise when resources use differing naming conventions, where there may be distinctions in terminology and taxonomies, where the same terms may be used inconsistently or have differing properties (such as different units of measure, enumerated values, or constraints). These mismatches are far more difficult to resolve than syntactic and structural incompatibilities, and necessitate costly, custom solutions at best. The approach promoted by most systems integrators today is to define a single, monolithic data model for use throughout an organization, which, while effectively eliminating semantic conflicts, either forces the organization to migrate all of its information to the new model or prohibits integration with autonomously developed sources. This approach is not always feasible due to cost, resource constraints, or business requirements to maintain legacy systems. Semantic characteristics are often implicitly represented in the data and are seldom well documented. Failing to address semantic or context- or content-specific integration issues may produce unsatisfactory results and may have a negative impact on systems integrity.

Natural language processing methods for the purposes of understanding and retrieving information from various static, dynamic, and interactive sources exist. These methods represent attempts to generalize (or refine) a set of query terms, to expand a set of query terms, or to construct computable queries from expressions used in normal human discourse. While these approaches support customization from the user's perspective, they do not support transparent querying or integration of disparate sources without requiring the user to have knowledge of the source specific terminology.

SUMMARY OF THE INVENTION

According to the present invention, there is provided a method of querying a plurality of data sources. A common query is received, the common query being constructed utilizing a common vocabulary that is mapped to a plurality of source vocabularies associated with respective data sources of the plurality of data sources. A plurality of source queries are generated to the plurality of data sources, each source query being derived from the common query and being constructed utilizing a respective source of vocabulary.

Other features of the present invention will be apparent from the accompanying drawings and from the detailed description that follows.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example and not limitation in the figures of the accompanying drawings, in which like references indicate similar elements and in which:

Figure 1 is a block diagram illustrating a prior art point-to-point integration scheme between a plurality of data consumers and a plurality of data sources.

Figure 2 is a block diagram illustrating the use of a common vocabulary to provide a two-level integration between a number of data consumers and a number of data sources, according to one embodiment of the present invention.

Figure 3 is a block diagram illustrating mappings between two common vocabularies and a plurality of source vocabularies, according to one embodiment of the present invention.

Figure 4A is a diagrammatic representation of an exemplary client-server (or two-tiered) environment within which the present invention may be deployed.

Figure 4B is a diagrammatic representation of an exemplary three-tier environment within which the present invention may be deployed.

Figure 5 is a block diagram illustrating architectural details of an exemplary data server constructed to implement one embodiment of the present invention.

Figure 6 is a flow chart illustrating the primary operations of an exemplary method of querying a plurality of data sources, and receiving a response to such a query, according to one embodiment of the present invention.

Figure 7 is a flow chart illustrating a method of generating a source query expression for an applicable source, the source query expression being derived from a common query expression, according to one embodiment of the present invention.

Figure 8 is a flow chart illustrating an exemplary method, according to one embodiment of the present invention, of performing attribute mapping function.

Figure 9 is a flow chart illustrating an exemplary method, according to one embodiment of the present invention, of performing a semantic mapping for algorithmic conversion of attribute values.

Figure 10 is a flow chart illustrating an exemplary method, according to one embodiment of the present invention, of performing semantic mapping for lookup-based mapping of attribute values.

Figure 11 is a diagrammatic representation of an exemplary attribute mapping table, according to one embodiment of the present invention.

Figure 12 is a block diagram illustrating an exemplary machine, in the form of a computer system, that may execute a sequence of instructions embodying the present invention and that includes a machine-readable medium for storing such a sequence of instructions.

DETAILED DESCRIPTION

Overview

A method and apparatus for querying a plurality of data sources are described. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will

be evident, however, to one skilled in the art that the present invention may be practiced without these specific details. The invention, in one embodiment, enables a distributed collection of applications and repositories with dissimilar syntactic and semantic characteristics to be accessed as though they are a single entity. Queries are performed transparently, without requiring the user to understand the characteristics of any individual source, through the use of a common vocabulary and semantic mappings to the source repositories. A common vocabulary is used to provide the desired transparency. Mappings establish the correspondence between the common vocabulary and each of the source-specific vocabularies which are to be integrated. The common vocabulary may be tailored to the needs of an individual or group of users and/or applications and forms the basis for resolution of syntactic and semantic conflicts and ambiguities within a federation of information resources. The terms and constraints defined by the common vocabulary are then used to construct the query. The terms in the common vocabulary are mapped to each applicable source vocabulary for query and retrieval purposes. By mapping the semantic characteristics of the various sources to the common vocabulary, the integrity of the individual source repositories, file systems, processes, and applications is maintained, while new applications and access methodologies can make use of the integrated resources.

A method of using semantic vocabularies and mappings is described. This approach allows data systems to be integrated or accessed transparently, from a single point of entry, despite differences in the semantic characteristics of the applications or repositories.

In one exemplary embodiment, the requisite vocabularies and semantic mappings may be prepared and information sources may be integrated using the following operations:

(1) Each data source to be integrated is modeled to create a source vocabulary. The vocabulary is essentially a structural, syntactic and semantic schema for the source.

(2) A common vocabulary, containing any combination of terms and attributes from the source(s), and the relationship and constraints that formally define those terms, is created. The characteristics of the common vocabulary need not be the same [in structure, syntax, or semantics] as any of the individual source vocabularies. It should be noted that the terminology, formats and semantics of the common vocabulary may be determined by one or more domain-specific standards, organizational requirements, or the user's preference.

(3) The common vocabulary is mapped to each of the source vocabularies.

(4) The mappings are converted to a form that is usable by the run-time engine. This process will take the mapping between the names in the source vocabulary and variables in the common vocabulary (created in operation 3) and generate the contents of the mapping tables. The details for each vocabulary item are retrieved from the source models and the common models created in operations 1 and 2. The structural, syntactic and semantic correlations between a particular pair of vocabularies will be encoded. In the case of mappings performed by algorithms (*e.g.*, units-of-measure), it will be necessary determine if a mapping exists between the two forms. If not, this fact will be flagged for the knowledge engineer. In this case it is the responsibility of the knowledge engineer to create a mapping and update the list of mappings available to the knowledge base used by the generation process.

Allowing the vocabulary to be tailored enables the user to access and evaluate the information using familiar terminology and conventions. The use of a common vocabulary addresses the need for each unique client application to be integrated with each unique source. Furthermore, it allows the set of sources sharing this vocabulary to be queried transparently through a single query which need not be reformulated for each target repository. The user (and the client application) need not understand the specific details required to access each source. This approach allows sources to be integrated incrementally. Once a common vocabulary has been established, sources may be integrated at a pace determined by business needs. The common vocabulary can evolve over time as additional sources are integrated or new business requirements arise. Multiple common vocabularies, and thus multiple federated collections of applications and repositories, may be integrated using the same approach.

An exemplary use of the common vocabulary may be the semantic integration of legacy sources. A common vocabulary may also serve as a starting point from which new source systems are developed.

At run-time, the following operations are performed:

(1) A pre-existing common vocabulary is selected based on the problem domain, relevant business processes and requirements, the available source repositories and applications, and user preferences, as appropriate. The user (or application) may also select a subset of the available sources for searching through the use of a directory services mechanism.

(2) A query is formulated using the terms and constraints of the common vocabulary.

(3) Identical queries are distributed, in parallel, to each applicable source repository.

(4) Upon receipt, each server maps the parameters specified in the request from the common vocabulary to the source-specific vocabulary. This process may include term renaming, unit conversion, translation, or mapping attributes and values to those used by the source.

(5) The source application, database management system, file system, or other information processing system acts on the request and returns the result set (if any).

(6) The terms of the result set are mapped from the local vocabulary to the common vocabulary, and

(7) the result set is returned to the client for further processing.

As a mapping exists from each individual source vocabulary to the common vocabulary, queries may be performed transparently across the set of integrated sources. This arrangement allows modular, plug-in, incremental integration of new sources into an existing environment. By creating a common vocabulary for the existing system, new sources may be mapped to the common vocabulary without requiring modification of those sources. No changes are required to existing applications to allow a new source to be utilized. **Figure 1** illustrates the complexity of integration using the traditional, custom approach. The present invention is advantageous in that it may substantially reduce the number of point-to-point integrations needed among applications and repositories that comprise the user's environment. **Figure 3** illustrates the use of a common vocabulary 14 to resolve attribute names, data types and units between a number of sources 12 and a number of client (user) applications 10. Specifically, **Figure 3** illustrates the use of two independent, common vocabularies 14 which map combinations of attributes from three source vocabularies 16, each source vocabulary 12 being associated with a respective source 12. In one embodiment, each of the respective sources may display dissimilar structural syntax, semantics or access protocols from other source vocabularies 16.

The mapping component may be relocated to an application server servicing a cluster of sources 12 with minimal loss of flexibility. Whether vocabulary mapping functionality is performed by the information source server or a middle-tier application server, client applications (and therefore users) 10 are shielded from the specifics of the individual source vocabularies 16.

Exemplary Embodiment

Figure 4A illustrates an exemplary client-server environment 20 in which the present invention may be used. The environment 20 includes one or more client applications 10 which may utilize individual or shared vocabularies, and multiple data sources in the exemplary form of data servers 22 which may utilize individual or shared

vocabularies. Server vocabularies of the data servers 22 may overlap with the client vocabularies of the client applications 10 although this need not be the case. The data servers 22 may have dissimilar structure and syntax, dissimilar semantics and terminology, and dissimilar access protocols from the client applications 10. In one embodiment, at least one data server 22 supports one client vocabulary, and the client applications 10 and servers 22 are connected by a network 24 or other communication medium. The servers 22 (or a subset thereof) may share at least one common vocabulary 14. This common vocabulary 14 is utilized by at least one client application. There may be multiple common vocabularies. The term "common" is with respect to some set of data servers or sources.

Requests 26 are sent from the client application 10 to the server 22 and responses 28 from the servers 22 to the client application 10. The client application 10 generates requests 26 and receives responses 28 in the user's terminology and semantics (*i.e.*, the common vocabulary). Each data server 22 may receive requests 26 and generate responses 28 in the user terminology and semantics. However, the internal repositories utilizes the source-specific local terminology and semantics. It should be noted that a user terminology and semantics set is not arbitrary, rather it is a set of corresponding terms, concepts and preferences which have been tailored to one or more users. Each data server 22 sharing the common vocabulary 14 (or a user selected subset of the data servers 22) receives an identical request 26.

The exemplary architecture, illustrated in **Figure 4A**, is client-server. The present invention is equally applicable to a three-tiered architecture such as that shown in **Figure 4B**, where the mapping functionality is performed on (one or more) intermediate application servers 30. The application server 30 receives client requests, maps the requests and queries the appropriate data sources (e.g., the servers 22), then remaps and returns the results to the client. An application server 30 is also capable of manipulating the results, if necessary, to sort, merge or filter the results as well as performing report generation and analysis functions. The application server 30 understands the mappings between each common vocabulary and each source vocabulary accessible in the federation. In the client-server case illustrated in **Figure 4A**, each server 22 is required to understand only those mappings between supported common vocabularies and its local vocabulary. The application server 30 also supports all of the access protocols needed to access any of the data servers 22. By using a server-based process, the client-server architecture takes advantage of a common access protocol. Since the application server 30 intermediates each transaction, it may require relatively powerful hardware and may potentially become a bottleneck (depending on the number of users or the tasks performed on behalf of users) or single-point-of-failure. An application server 30 does, however, allow the mapping information and business applications to be centralized which may reduce the maintenance to clients and back-

end servers. Unlike the client-server architecture, the three-tier approach does not require a server-resident mapping process be added to each data server 22. Using a client-server or three-tiered architecture, the concept of integrating dissimilar information sources using a common vocabulary remains unchanged. It will be appreciated that, in the three-tier architecture illustrated in **Figure 4B**, certain of these components will be deployed on the application server 30.

Figure 5 shows the primary components of an exemplary data server 22 as displayed within the two-tier (client-server) architecture shown in **Figure 4A**. The illustrated primary components of the exemplary data server 22 include a communication interface 32, a mapper 34 and a source interface 36 (*e.g.*, a user API, a DBMS) to the information repositories 38. The communication interface 32 is arbitrary and supports whatever communication methodologies (*e.g.*, TCP/IP, CORBA, ATM, Ethernet) is required to communicate over the network 24. The communication interface 32 also provides services to format/extract information to/from messages. As previously discussed, these messages utilize the users' terminology and semantics set. As illustrated in **Figure 5**, the mapper 34 transforms the users' terminology and semantics embodied in a request 26 into the local terminology and semantics (and vice versa). Once translated into the source-specific local terminology, a query is made to the appropriate data source 38 (e.g., database, file or other data management service). In order for the mapper 34 to perform the necessary translations, a number of mapping tables 40 are made available to it.

Since the client application 10 has knowledge of only the common vocabulary 14, the capability of bi-directional mapping is provided. The top half of **Figure 5** shows the mapping requests 26 from common to source vocabulary, while the lower half shows the mapping responses 28 from source to common vocabulary.

While not required for the basic functionality, a variable length string may be used to transfer the requests and responses between the client application 10 and the data server 22. In the data server 22, lexical tokens, representing attribute names, operators and values may be extracted by a lexical analyzer 42, after which the mapping appropriate to the token (*e.g.*, name or value) is performed. A syntax analyzer 44 determines whether a token is an attribute name, an operator, a value or something else. An alternative is to remember what the last token was (assuming the use of <attribute_name, operator, attribute_value> triples). The mapping functions may be built as actions to the lexical analyzer 42 and generation of the query expression may be performed by actions in the syntax analyzer 44. Relational expressions may be chained together in compound expressions as shown in **Figure 7**. The input and output grammars are chosen by the user according to the application. The lexical analyzer 42 and syntax analyzer 44 skeletons may be generated with commercially available tools

used for that purpose (e.g., AT&T lex and yacc). These skeletons may then augmented with the appropriate set of actions.

Figure 6 is a flow chart illustrating an overview of an exemplary method 50 of performing a mapping of a request 26, expressed in a common vocabulary, to a source vocabulary 16. The method 50 commences with lexical extraction operation 52 by the lexical analyzer 42, followed by a semantic mapping operation 54 performed by the mapper 34 and a syntax analysis operation 56 performed by the syntax analyzer 44.

Two types of semantic mapping are performed as part of the semantic mapping operation 54 in order to provide the desired transparency, namely attribute name mapping 60 and attribute value mapping 62. Attribute name mapping 60, further illustrated in **Figure 8**, resolves naming differences between the attributes in the vocabularies. Attribute value mapping 62 resolves differences between the contents of the attributes. Attribute value mapping 62 may be performed in one of two exemplary methods: algorithmically, as illustrated in **Figure 9**, or by lookup, as illustrated in **Figure 10**.

Figure 7 is a flow chart illustrating further details of a method 66, according to one embodiment of the present invention, of expression generation. It should be noted that **Figure 7** does not illustrate lexical, syntactic or semantical analysis performed through compiler functionality, nor symbol table management or generation provided by compiler technology or techniques.

The method 66 illustrates the chaining together of relational expressions into a compound expression. The method 66 commences with a mapping operation 54, followed by a determination at decision block 68 as to whether a logical operation has been detected. If so, the method 66 continues with the logical expression formation by proceeding via block 70 to perform a further mapping operation 54. On the other hand, should no logical operation be detected at decision box 68, a compound expression, comprising a chained series of relation expressions, is submitted at block 72.

As stated above, two types of semantic mapping are performed as part of the semantic mapping operation. **Figure 8** is a flow chart illustrating a method 71, according to an exemplary embodiment of the present invention, of performing an attribute name mapping 60.

The method 71 commences at block 72 with a lookup attribute name operation. Specifically, a global name 74 is utilized to perform a lookup with respect to an attribute name table 78, and to return a local name 76.

As shown in **Figure 11**, there should be at least one entry within the attribute name table 78 for each attribute which stores the mapped global name 74, the mapped data type and a string which specifies any required algorithmic conversion(s). A key consisting of the vocabulary name, the class name and the attribute (or local) name 76 concatenated together is used as a key to a database lookup. This key structure allows

multiple vocabularies to be stored in a single database or table. It also allows the same attribute name 76 to exist in different classes as well as for the same class name to exist in different vocabularies. Once the mapping information is retrieved, the attribute (local) name 76 found in the attribute name table 78 is written to an output string followed by the detected relational operator.

Returning to **Figure 8**, at decision box 80, a determination is made as to whether the local name is valid for a relevant source vocabulary 16 (i.e., does the local name correspond to an actual name within the source vocabulary 16). If not, an error is generated at block 82. Alternatively, at block 84, the local name 76 is inserted into the query expression.

At block 86, the attribute data type (e.g., floating point number, string or integer) is retrieved. At block 88, formatting of the expression commences.

The lookup attribute value mapping 62 also includes two components, namely (1) a semantic mapping for algorithmic conversion component and (2) a semantic mapping for lookup-based mapping of attribute values component.

Figure 9 is a flow chart illustrating an exemplary method 90, according to one embodiment of the present invention, of performing a semantic mapping of algorithmic conversions on attribute values.

At block 92, upon the detection of a value token (or operand), the lexical type for the relevant value token is determined.

In a first instance, if the value associated with the value token is typed as a floating point number, an algorithmic conversion, if indicated by the mapping structure, is performed at block 94. Similarly, if the value associated with the value token is typed as an integer, an algorithmic conversion, if indicated by the mapping structure, is performed at block 96. To perform the algorithmic conversions at blocks 94 and 96, the mapper 34 has access to a library of supported conversions integrated into it.

Examples of algorithmic conversions that may be performed at blocks 94 and 96 include unit-of-measure conversions (e.g., centimeters to inches) or any other conversions of values that may be algorithmically performed, as opposed to requiring a lookup operation.

Figure 10 is a flow chart illustrating an exemplary method 100, according to one exemplary embodiment of the present invention, of performing a semantic mapping for lookup-based mapping of attribute values.

As noted, if a value is typed as a floating point number at block 92 in **Figure 9**, is subject to an algorithm conversion at block 94. **Figure 10** reveals that the value receives no further value mapping, and is accordingly inserted into an output string and a next token value is located.

On the other hand, for string or integer typed values, additional lookups may be performed. A key (not shown) consisting of a vocabulary name, a class name, an attribute name and the value are concatenated together and utilized as a key to a database lookup in the name table 78 or a value table 79.

Dealing first with a string typed value, two attempts may be made to map such a string value. First, at block 102, a lookup is made in the attribute name table 78 to determine whether or not the value is in fact another attribute (local) name. If the lookup at block 102 succeeds, as determined at decision block 104, the local name 76 substitutes the global name 74, the result is placed in an output string, and the next value token is sorted.

On the other hand, should it be determined at decision block 104 that the lookup at block 102 has failed, the global name 74 is retained at block 108 and the key is utilized to perform a second lookup at block 110 in an attribute value table 79 to determine if the key is to an enumerated value.

If the lookup operation at block 110 succeeds, as determined at decision block 112, then a the local value retrieved from the value table 79 is substituted for the global value, written to an output string, whereafter the next value token is sought.

On the other hand, should it be determined at decision block 112 that the lookup operation at block 110 has failed, then the original, global value is retained at block 116, and written into an output string. Thereafter, the next value token is sought.

In the case where the value token is determined at block 92 to comprise an integer type value, following any required algorithmic conversions at block 96, the actions described above with reference to blocks 110 - 116 are then performed with respect to such an integer type value.

It should be noted that the two stage mapping of string values is only needed for applications involving query generation (since the value of two attributes may be compared with one another). In an alternative embodiment, only a single lookup may be used. Specifically, if a value token representing a logical operator is located, the logical operator will be added to an output string, and the process repeated. When the end of a valid expression is detected, the output string is used to query the data source. The results returned from the data source are then mapped back into the common vocabulary 14. The reverse mapping process is substantially as described above, with two exceptions. First, the mapping tables (e.g., the name table 78 and the value table 79) are from the source (or local) vocabulary 16 to the common vocabulary 14. Secondly, the input and output are <attribute_name, attribute_value> tuples.

Figure 11 shows the structure of exemplary mapping tables 64 within the context of an object-oriented database. The same structure may be implemented for the attribute name and the attribute value tables 78 and 79. In the attribute name table 78, all three fields are needed. For the attribute value table 79, only the field for the

mapped value field is needed. A mapped name may furthermore be an attribute name or an attribute value. The tables 64 may be implemented through the object-oriented database using a hashed dictionary structure 130. Another data structure may be used so long as it has a look-up or search capability. The data structures, itself, need not be persistent, if it can be loaded from a persistent source (*e.g.*, a file) when required. A database does, however, provide built-in mechanisms to more easily manage, store, retrieve and update the data. It is clear that the choice of data structures may affect the performance of the mapping functions. The hashed dictionary structure 130 generally provides fast lookup performance relative to other commonly used methods if an effective hash function is selected. Since these tables 64 are of the same format and the key string allows sufficient uniqueness, the two tables may be merged into one. This approach, however, may not significantly reduce the number of lookups, the aggregate table size, or the maintenance requirements but may result in some loss of understandability.

As noted above, in order to perform the required algorithmic conversions, the mapper 34 has a library of supported conversions integrated into it. One embodiment of the present invention links a library (static or dynamic) of conversion algorithms into the mapper 34. The appropriate conversion function is selected based on a code in the conversion specifier string, as shown in **Figure 11**. These may be packaged as remote procedures or distributed functions such as the Object Management Group's Common Object Request Broker Architecture (CORBA). This integration allows new functions to be easily added.

Figure 12 is a block diagram illustrating a machine, in the exemplary form of a computer system 250, within which a set of instructions for causing the computer system 250 to perform any one of the methodologies discussed above may be executed. The computer system 250 includes a processor 252, a main memory 254, and a static memory 255, which communicate with each other via a bus 256. The computer system 250 further includes a video display unit 258 (e.g., a liquid crystal display (LCD) or a cathode ray tube (CTR)). The computer system 250 further includes an alpha-numeric input device 260 (e.g., a keyboard), a cursor control device 262 (e.g., a mouse), a storage medium in the exemplary form of a disk drive unit 264, a signal generation device 266 (e.g., a speaker) and a network interface device 268.

The disk drive unit 264 includes a machine-readable medium 265 on which is stored a set of instructions (i.e., software 270) embodying any one, or all, of the methodologies described above. The software 270 is also shown to reside, completely or at least partially, within the main memory 254 and/or within the processor 252. The software 270 may furthermore be transmitted or received via the network interface device 268. For the purposes of the present specification, the term "machine-readable medium" shall be taken to include any medium that is capable of storing and encoding

a sequence of instructions for execution by the machine, and that causes the machine to perform any one of the methodologies of the present invention. The term "machine-readable medium" shall accordingly be taken to include, but not limited to, solid-state memories, optical and magnetic disks, and carrier wave signals.

The present application is useful for developing and integrating applications and repositories wherein the data is accessed using terminology, methodologies, or other mechanisms that may differ from the source implementation, format, or context. It may be used to resolve both semantic and syntactic differences among applications and repositories. Although databases are used as examples throughout this disclosure, the approach is independent of the source implementation or underlying technology. This invention is also applicable to information brokering and mediation.

The present invention is also advantageous in that data migration is not required to customize the vocabulary or transparently to query multiple data sources as a single set. This alone results in a significant reduction of cost, operational disruption and risk. As the integrity of the various source repositories and applications is preserved, the operation of existing systems remains uninterrupted while new applications can access and make use of the information maintained by those systems.

While an exemplary embodiment of the present invention has been described above in the context of a query that is targeted to a set of structurally, syntactically or semantically diverse set of information sources, it will be appreciated that the invention extends to the integration of information sources through the use of a common vocabulary to resolve structural, syntactic or semantic differences between subsources and the customization and resolving of an interface to a set of structurally, syntactically or semantically diverse set of data sources utilizing terminology particular to a user's preference or task area.

Thus, a method and apparatus for querying a plurality of data sources have been described. Although the present invention has been described with reference to specific exemplary embodiments, it will be evident that various modifications and changes may be made to these embodiments without departing from the broader spirit and scope of the invention. Accordingly, the specification and drawings are to be regarded in an illustrative rather than a restrictive sense.

CLAIMS

What is claimed is:

1. A method of querying a plurality of data sources, the method including:

receiving a common query constructed utilizing a first common vocabulary, the first common vocabulary being mapped to a plurality of source vocabularies associated with respective data sources of the plurality of data source; and

generating a plurality of source queries to the plurality of data sources, each source query being derived from the common query and being constructed utilizing a respective source vocabulary.

2. The method of claim 1 wherein the common query is constructed utilizing a first term of the first common vocabulary, the first term of the first common vocabulary being mapped by at least one mapping structure to corresponding first terms of the plurality of source vocabularies associated with the respective data sources.

3. The method of claim 2 wherein the generating of the plurality of source queries includes identifying the corresponding first terms of the plurality of source vocabularies utilizing the first term of the first common vocabulary and the at least one mapping structure, and including the first terms of the plurality of source vocabularies within the respective source queries directed to the plurality of data sources.

4. The method of claim 1 wherein the first common vocabulary is semantically mapped to the plurality of source vocabularies so as to address semantic differences across the plurality of source vocabularies.

5. The method of claim 1 wherein the common vocabulary is syntactically mapped to the plurality of source vocabularies so as to address syntactic differences across the plurality of source vocabularies.

6. The method of claim 1 wherein the common vocabulary is structurally mapped to the plurality of source vocabularies so as to address structural differences across the plurality of source vocabularies.

7. The method of claim 1 wherein the generating of the plurality of source queries comprises performing a name translation for the first term between a common name and a source name.

8. The method of claim 1 wherein the generating of the plurality of source queries comprises performing a value translation for a value associated with the first term between a common value and a source value.

9. The method of claim 8 wherein the value translation comprises a mapping of enumerated variables.

10. The method of claim 8 wherein the value translation comprises an algorithmic conversion of the common value to the source value.

11. The method of claim 1 wherein the first common vocabulary comprises terms specific to a first entity and a second common vocabulary comprises terms specific to a second entity.

12. Apparatus for distributing a common query across a plurality of data sources, the apparatus including:

a communication interface to receive a global query constructed utilizing a global vocabulary, the global vocabulary being mapped to a plurality of local vocabularies associated with respective data sources of the plurality of data sources; and

a mapper to generate a plurality of local queries to the plurality of data sources, each local query being derived from the global query and being constructed utilizing a respective local vocabulary.

13. The apparatus of claim 12 wherein the global query is constructed utilizing a first term of the global vocabulary, and wherein the mapper is to map the first term of the global vocabulary to at least one corresponding first term of a local vocabulary associated with a data source.

14. The apparatus of claim 13 wherein the mapper is to identify the first term of the source vocabulary utilizing the first term of the global vocabulary, and includes the first term of the local vocabulary within a respective local query directed to the data source.

15. The method of claim 12 wherein the mapper is to map the global vocabulary to at least one local vocabulary so as to address semantic specifics of the at least one local vocabulary.

16. The apparatus of claim 12 wherein the mapper is to map the global vocabulary to at least one local vocabulary so as to address syntactic specifics of the at least one local vocabulary.

17. The apparatus of claim 12 wherein the mapper is to map the global vocabulary to at least one local vocabulary so as to address structural specifics of the at least one local vocabulary.

18. The apparatus of claim 13 wherein the mapper is to perform a name translation of the first term between a global name and a local name.

19. The apparatus of claim 13 wherein the mapper is to perform a value translation for a value associated with the first term between a common value and a source value.

20. The apparatus of claim 19 wherein the mapper is to perform a mapping of enumerated variables.

21. The apparatus of claim 19 wherein the mapper is to perform an algorithmic conversion between a global value and a local value.

22. Apparatus for distributing a common query across a plurality of data sources, the apparatus including:

first means for receiving a global query constructed utilizing a global vocabulary, the global vocabulary being mapped to a plurality of local vocabularies associated with respective data sources of the plurality of data sources; and

second means for generating at least one local query to a data source, the local query being derived from the global query and being constructed utilizing the respective local vocabulary.

23. A machine-readable medium storing a sequence of instructions that, when executed by a machine, cause the machine to:

receive a common query constructed utilizing a first common vocabulary, the first common vocabulary being mapped to a plurality of source vocabularies associated with respective data sources of the plurality of data source; and

generate a plurality of source queries to the plurality of data sources, each source query being derived from the common query and being constructed utilizing a respective source vocabulary.

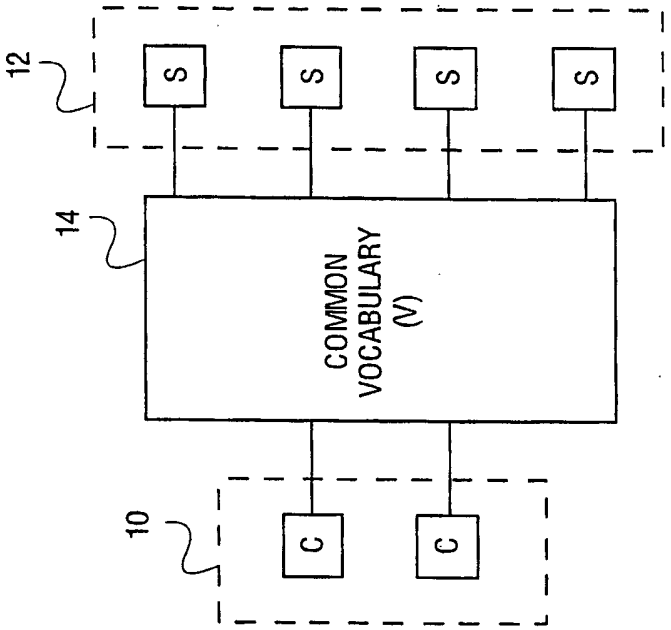


FIG. 2

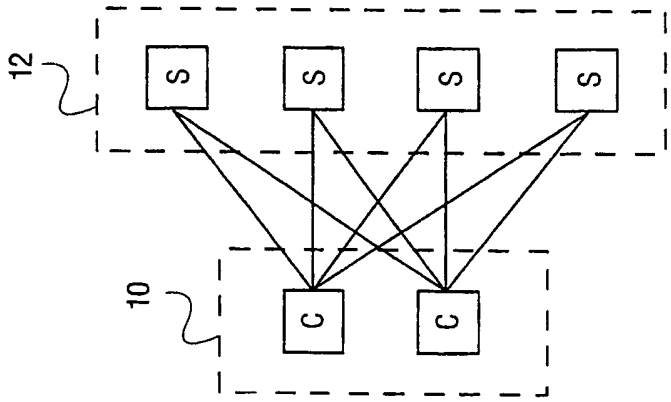


FIG. 1
(PRIOR ART)

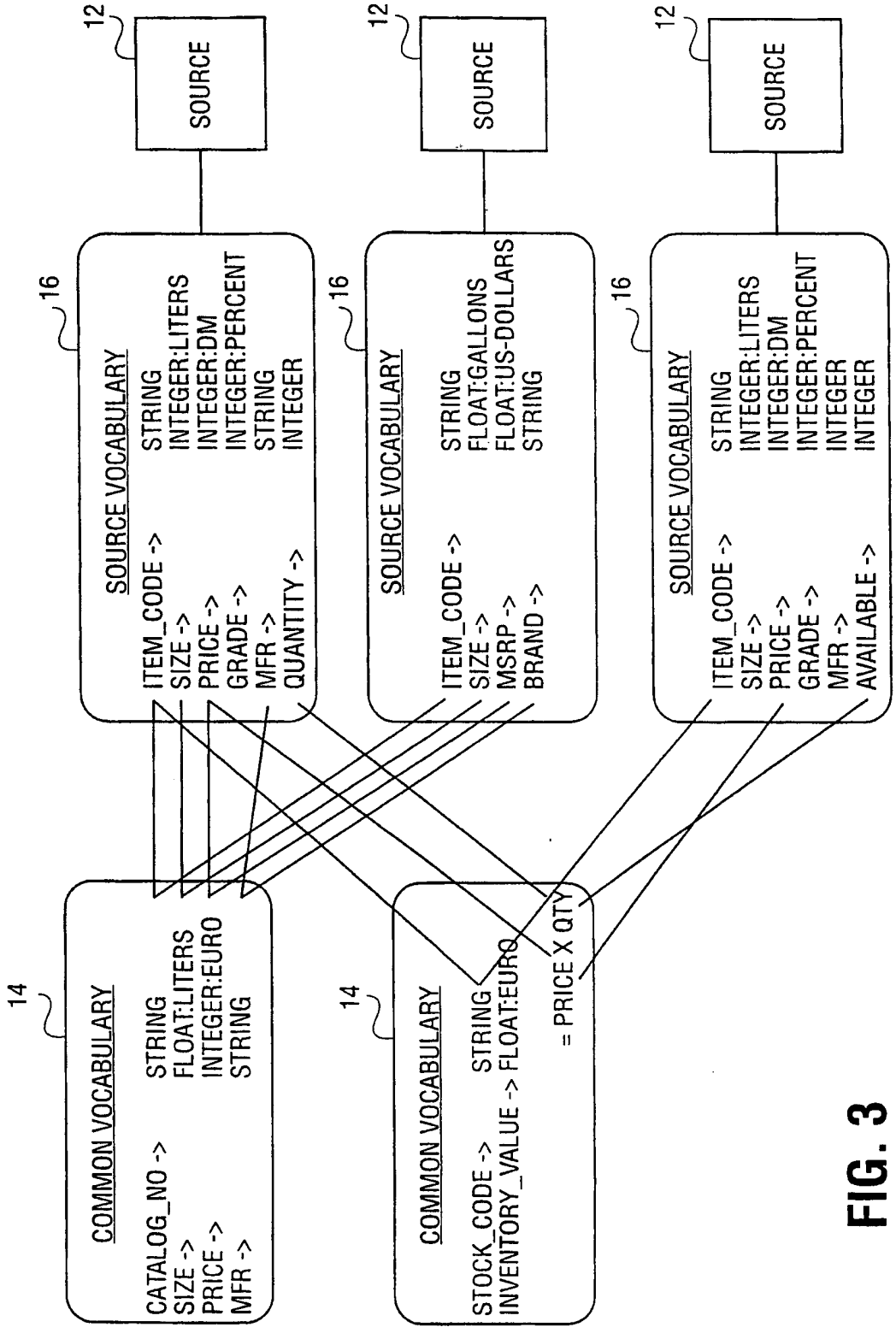

12
SOURCE
12
SOURCE
12
SOURCE
16
SOURCE VOCABULARY
ITEM_CODE -> STRING
SIZE -> INTEGER:LITERS
PRICE -> INTEGER:DM
GRADE -> INTEGER:PERCENT
MFR -> STRING
QUANTITY -> INTEGER
16
SOURCE VOCABULARY
ITEM_CODE -> STRING
SIZE -> FLOAT:GALLONS
MSRP -> FLOAT:US-DOLLARS
BRAND -> STRING
16
SOURCE VOCABULARY
ITEM_CODE -> STRING
SIZE -> INTEGER:LITERS
PRICE -> INTEGER:DM
GRADE -> INTEGER:PERCENT
MFR -> INTEGER
AVAILABLE -> INTEGER
14
COMMON VOCABULARY
CATALOG_NO -> STRING
SIZE -> FLOAT:LITERS
PRICE -> INTEGER:EURO
MFR -> STRING
14
COMMON VOCABULARY
STOCK_CODE -> STRING
INVENTORY_VALUE -> FLOAT:EURO
= PRICE X QTY

FIG. 3

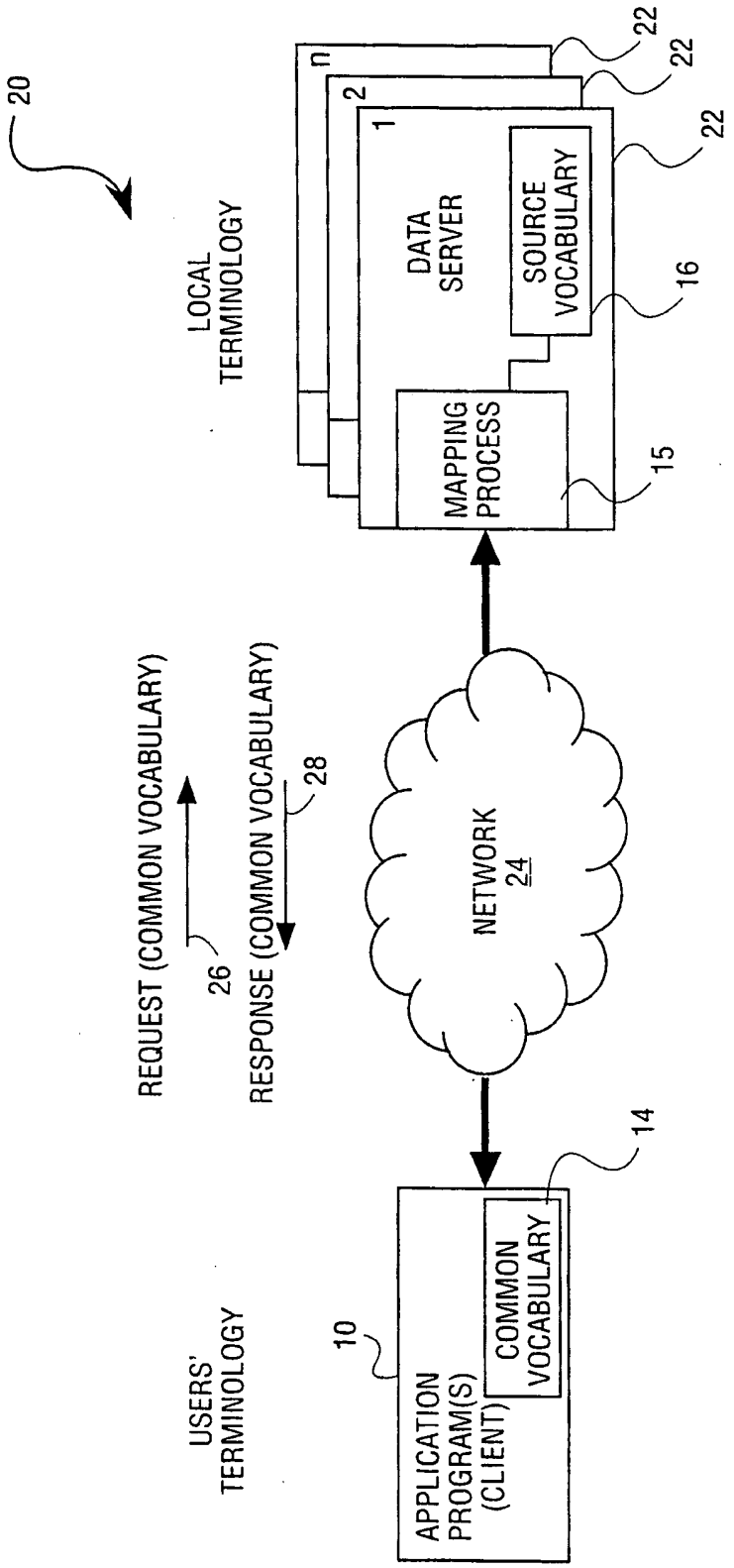

20
LOCAL TERMINOLOGY
n
2
1
22
22
22
DATA SERVER
SOURCE VOCABULARY
16
MAPPING PROCESS
15
NETWORK 24
REQUEST (COMMON VOCABULARY)
26
RESPONSE (COMMON VOCABULARY)
28
USERS' TERMINOLOGY
10
APPLICATION PROGRAM(S) (CLIENT)
COMMON VOCABULARY
14

FIG. 4A

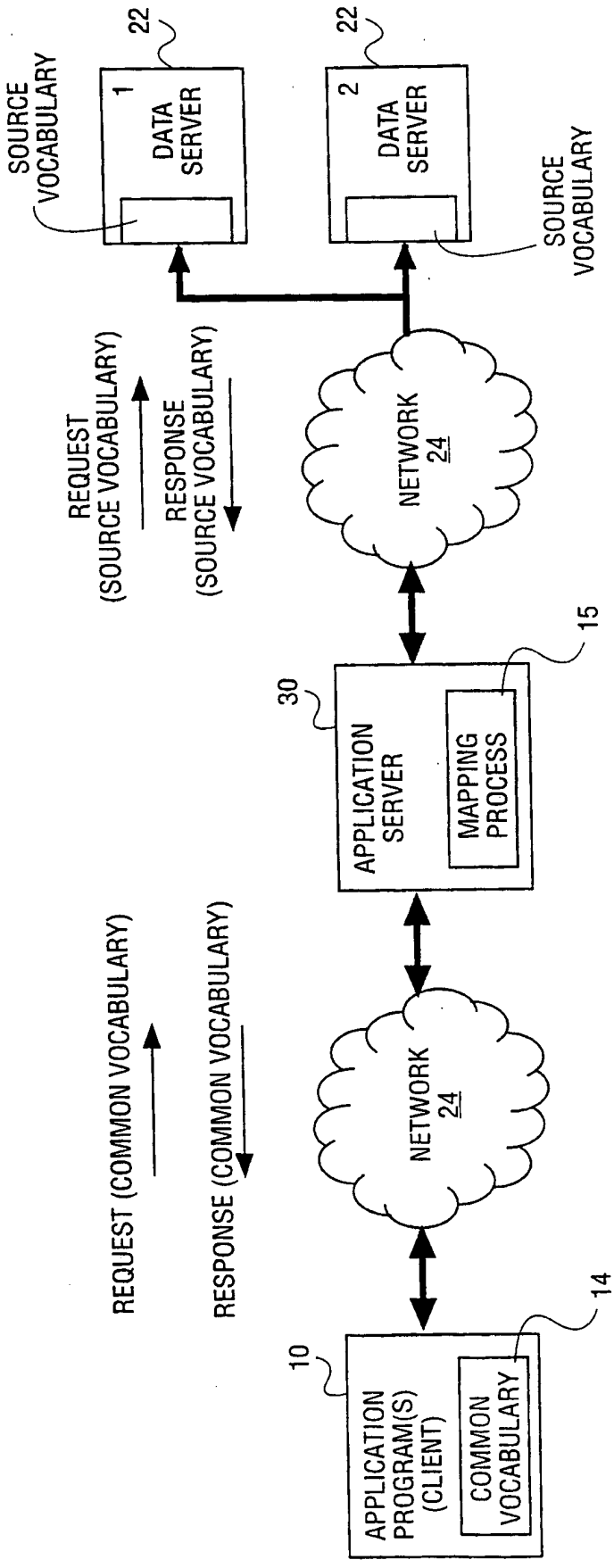

22
1
DATA SERVER
SOURCE VOCABULARY
22
2
DATA SERVER
SOURCE VOCABULARY
REQUEST (SOURCE VOCABULARY)
RESPONSE (SOURCE VOCABULARY)
NETWORK 24
15
30
APPLICATION SERVER
MAPPING PROCESS
REQUEST (COMMON VOCABULARY)
RESPONSE (COMMON VOCABULARY)
NETWORK 24
14
10
APPLICATION PROGRAM(S) (CLIENT)
COMMON VOCABULARY

FIG. 4B

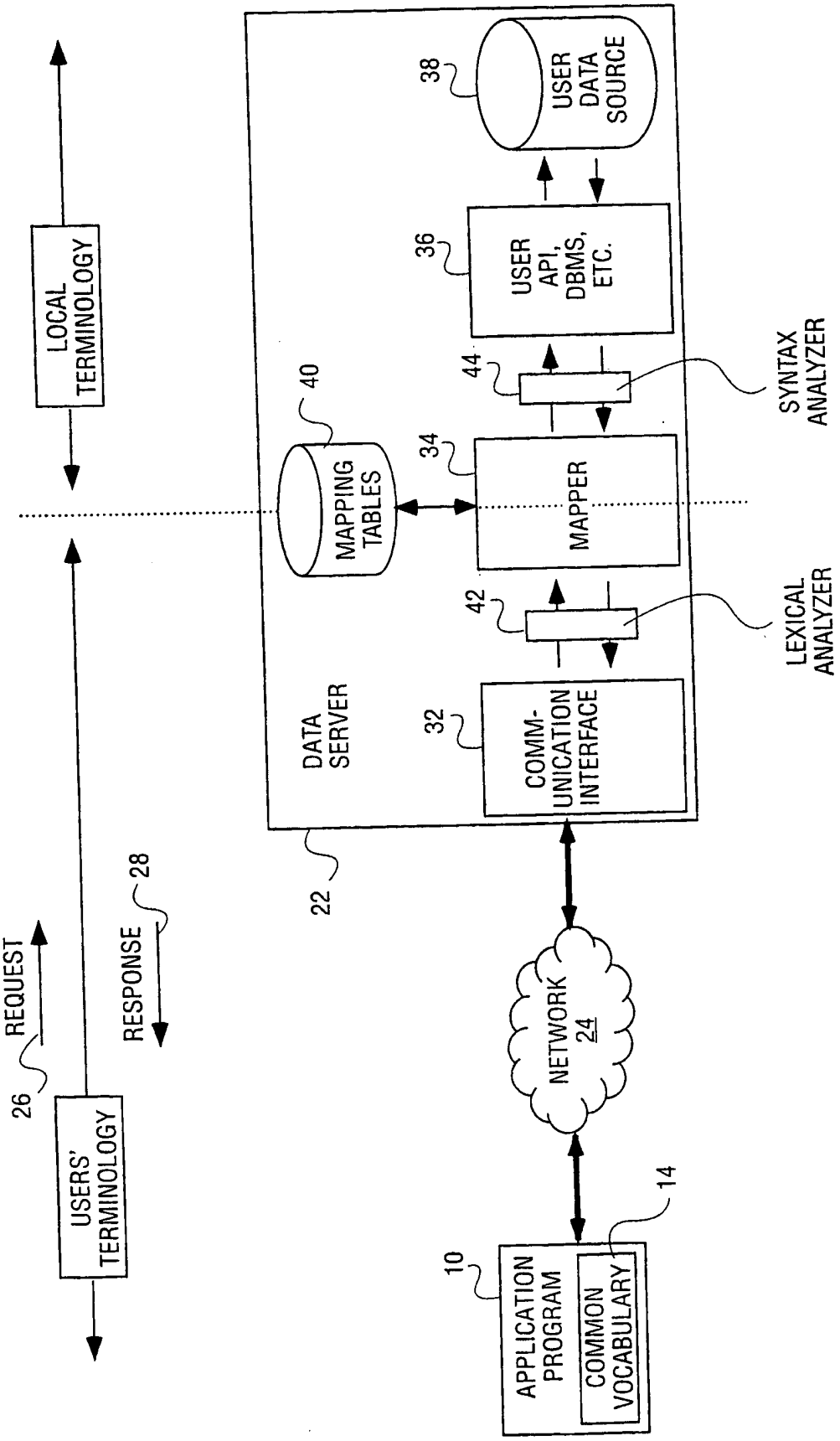

LOCAL TERMINOLOGY
USERS' TERMINOLOGY
REQUEST
RESPONSE
26
28
DATA SERVER
22
MAPPING TABLES
40
MAPPER
34
USER API, DBMS, ETC.
36
USER DATA SOURCE
38
SYNTAX ANALYZER
44
LEXICAL ANALYZER
42
COMM-UNICATION INTERFACE
32
NETWORK 24
APPLICATION PROGRAM
10
COMMON VOCABULARY
14

FIG. 5

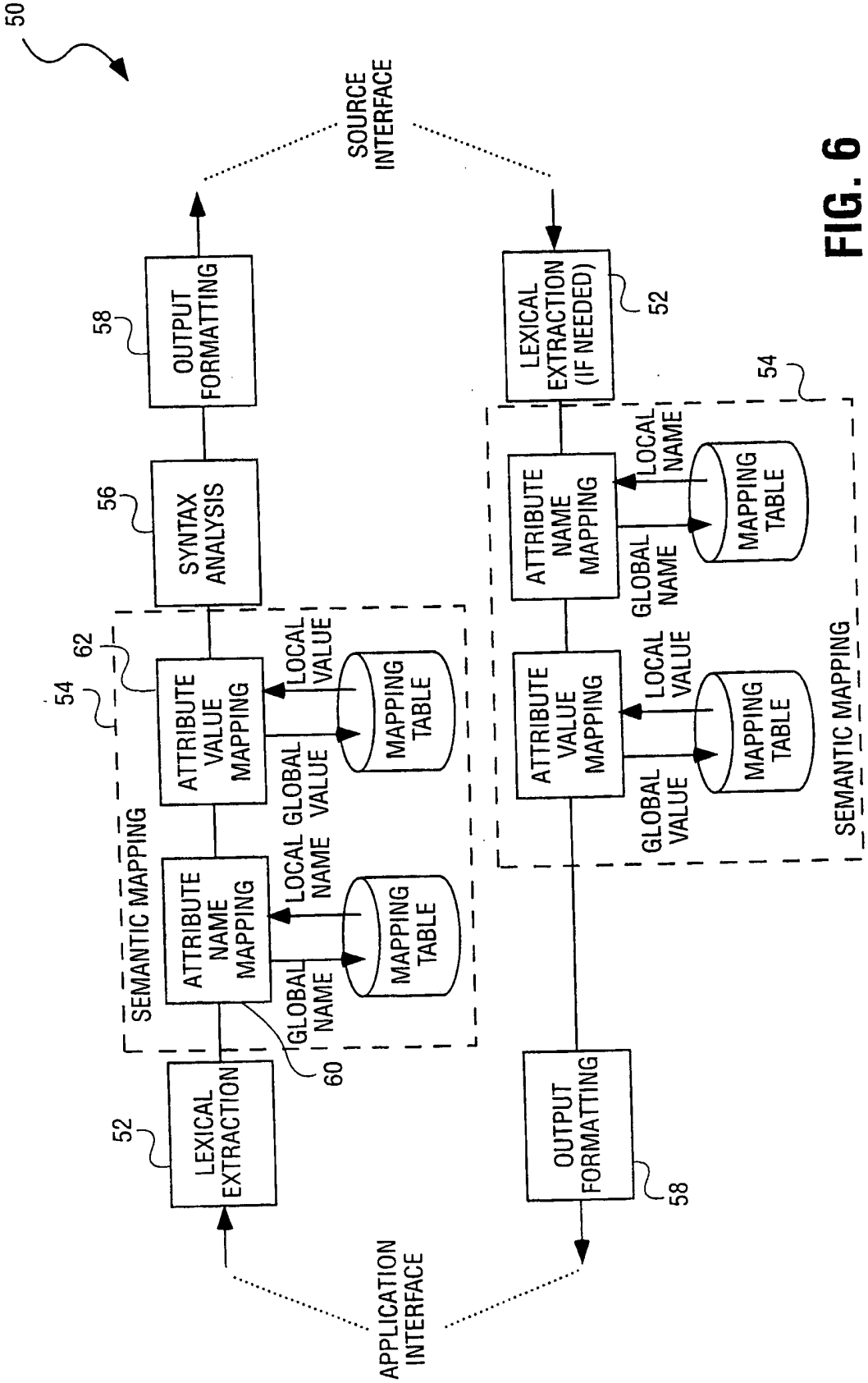

50
SOURCE INTERFACE
APPLICATION INTERFACE
LEXICAL EXTRACTION
52
LEXICAL EXTRACTION (IF NEEDED)
52
OUTPUT FORMATTING
58
OUTPUT FORMATTING
58
SYNTAX ANALYSIS
56
SEMANTIC MAPPING
54
SEMANTIC MAPPING
54
ATTRIBUTE NAME MAPPING
ATTRIBUTE NAME MAPPING
ATTRIBUTE VALUE MAPPING
62
ATTRIBUTE VALUE MAPPING
LOCAL NAME
GLOBAL NAME
LOCAL VALUE
GLOBAL VALUE
MAPPING TABLE
MAPPING TABLE
MAPPING TABLE
MAPPING TABLE
60

FIG. 6

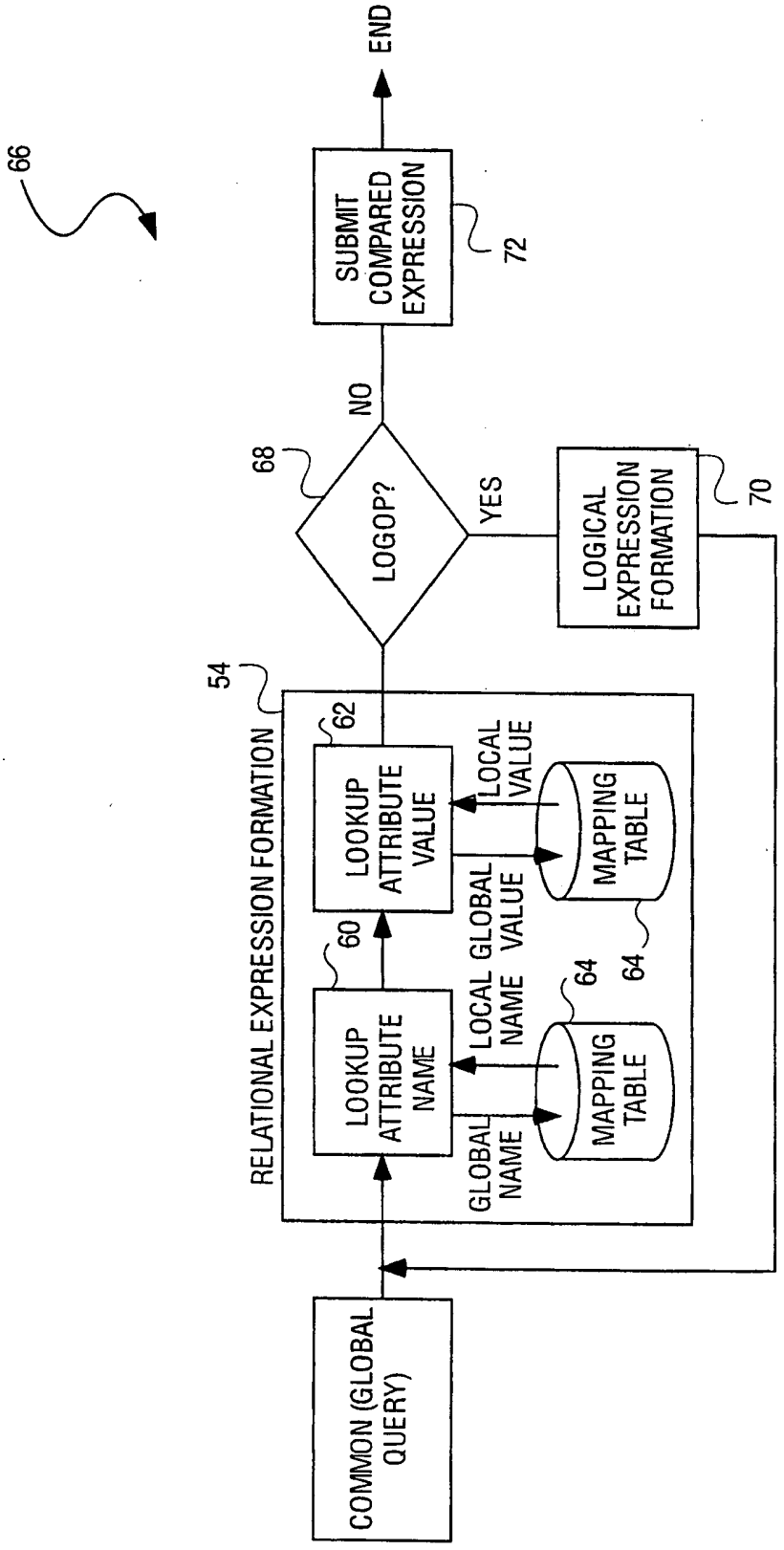

66
COMMON (GLOBAL QUERY)
RELATIONAL EXPRESSION FORMATION
54
LOOKUP ATTRIBUTE NAME
60
LOOKUP ATTRIBUTE VALUE
62
GLOBAL NAME
LOCAL NAME
MAPPING TABLE
64
LOCAL VALUE
GLOBAL VALUE
MAPPING TABLE
64
LOGOP?
68
YES
NO
LOGICAL EXPRESSION FORMATION
70
SUBMIT COMPARED EXPRESSION
72
END

FIG. 7

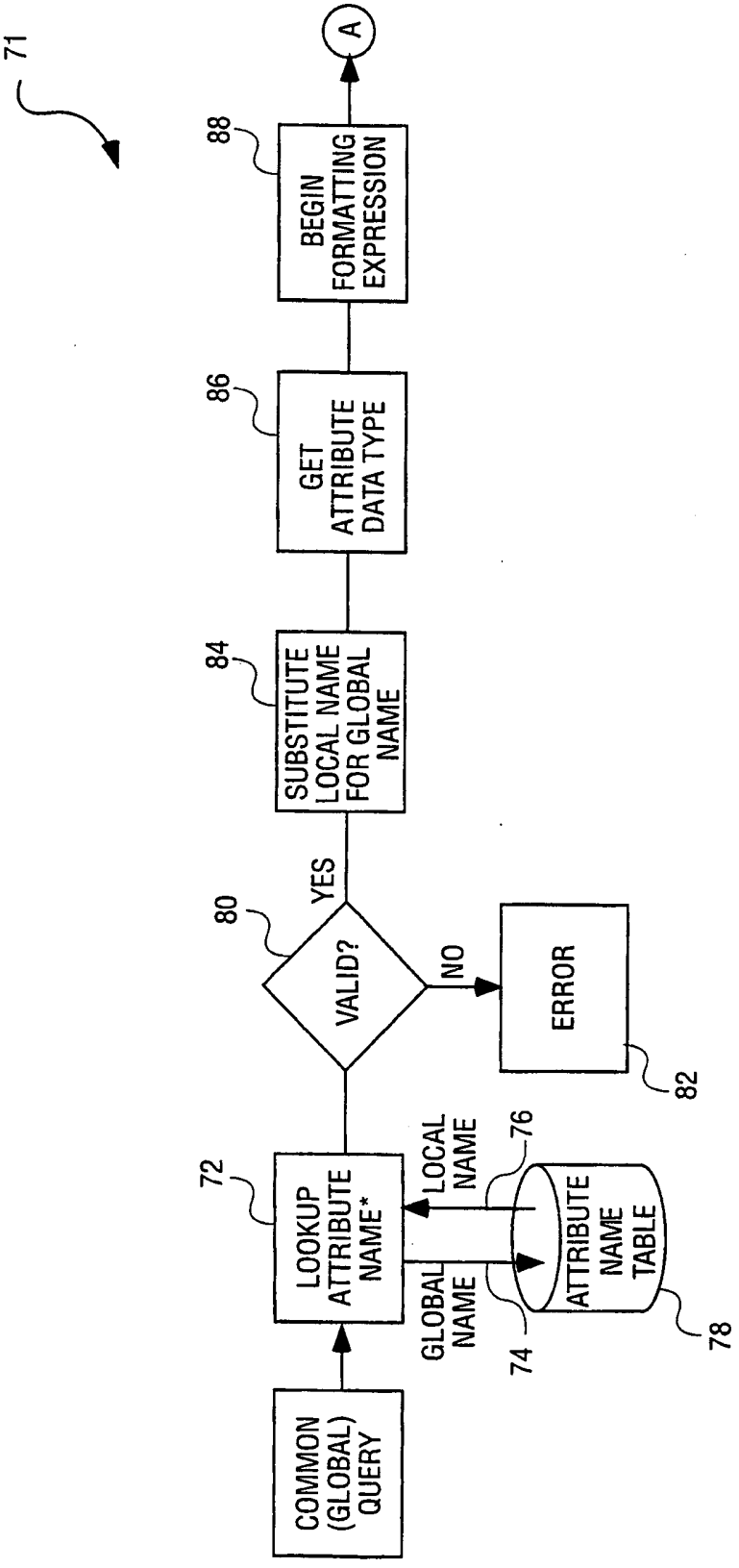

71
A
88
BEGIN FORMATTING EXPRESSION
86
GET ATTRIBUTE DATA TYPE
84
SUBSTITUTE LOCAL NAME FOR GLOBAL NAME
80
VALID?
YES
NO
ERROR
82
72
LOOKUP ATTRIBUTE NAME*
LOCAL NAME
76
GLOBAL NAME
74
ATTRIBUTE NAME TABLE
78
COMMON (GLOBAL) QUERY

FIG. 8

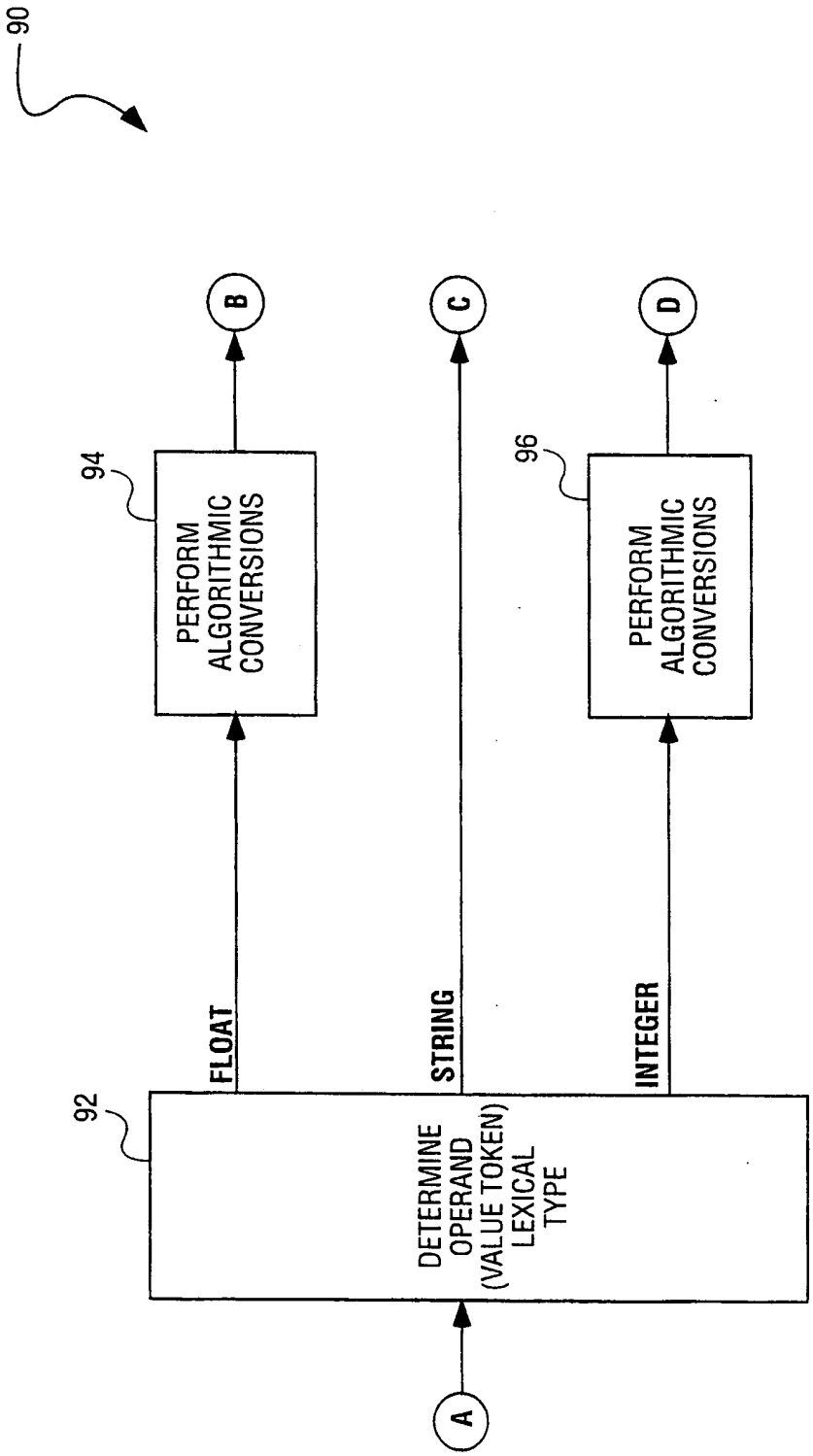

90
92
DETERMINE OPERAND (VALUE TOKEN) LEXICAL TYPE
FLOAT
STRING
INTEGER
94
PERFORM ALGORITHMIC CONVERSIONS
96
PERFORM ALGORITHMIC CONVERSIONS
A
B
C
D

FIG. 9

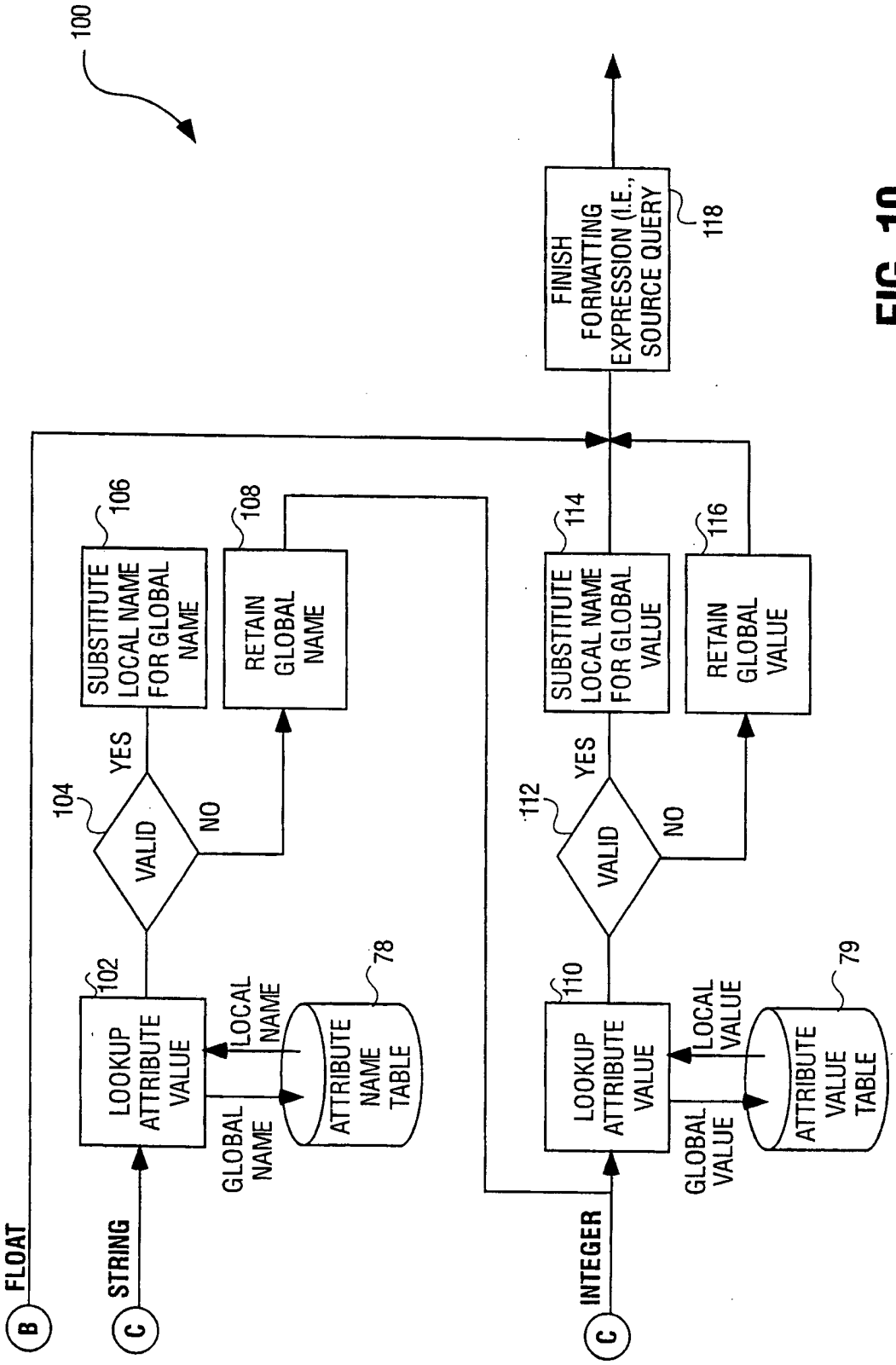

100
FLOAT
STRING
INTEGER
LOOKUP ATTRIBUTE VALUE
102
LOCAL NAME
GLOBAL NAME
ATTRIBUTE NAME TABLE
78
VALID
104
YES
NO
SUBSTITUTE LOCAL NAME FOR GLOBAL NAME
106
RETAIN GLOBAL NAME
108
LOOKUP ATTRIBUTE VALUE
110
LOCAL VALUE
GLOBAL VALUE
ATTRIBUTE VALUE TABLE
79
VALID
112
YES
NO
SUBSTITUTE LOCAL NAME FOR GLOBAL VALUE
114
RETAIN GLOBAL VALUE
116
FINISH FORMATTING EXPRESSION (I.E., SOURCE QUERY
118

FIG. 10

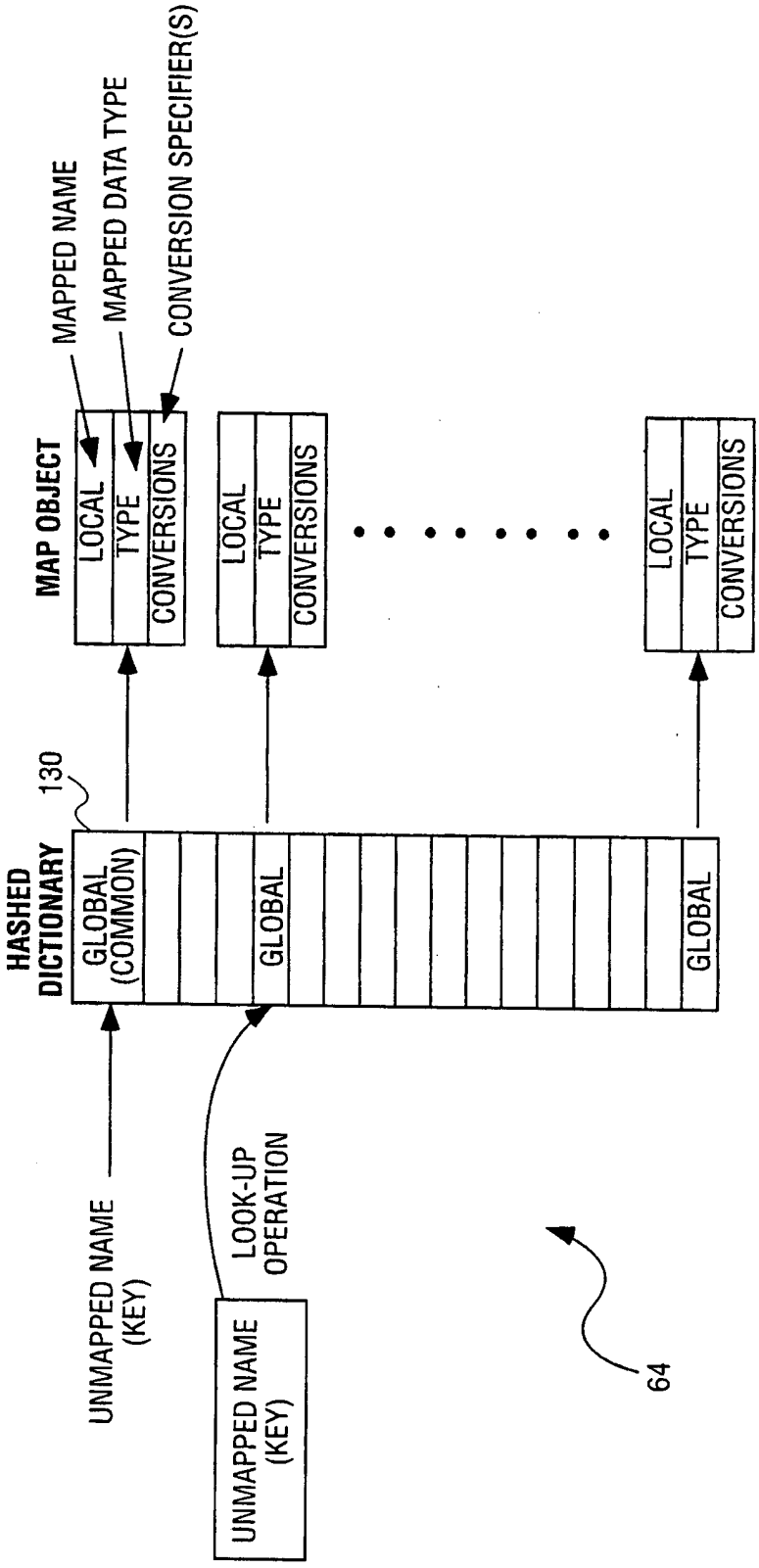

MAPPED NAME
MAPPED DATA TYPE
CONVERSION SPECIFIER(S)
MAP OBJECT
LOCAL
TYPE
CONVERSIONS
LOCAL
TYPE
CONVERSIONS
LOCAL
TYPE
CONVERSIONS
130
HASHED DICTIONARY
GLOBAL (COMMON)
GLOBAL
GLOBAL
UNMAPPED NAME (KEY)
LOOK-UP OPERATION
UNMAPPED NAME (KEY)
64

FIG. 11

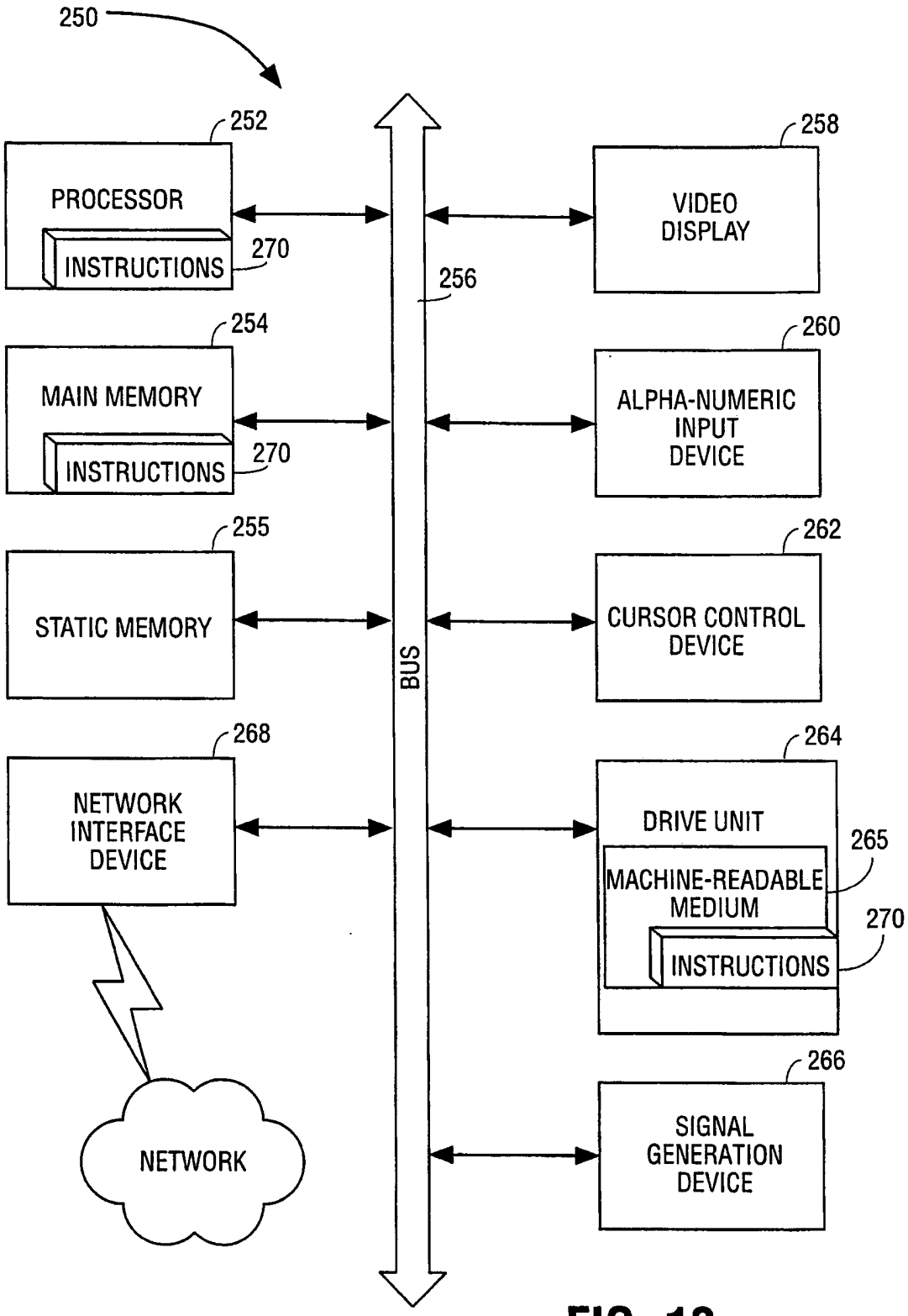

250
252
PROCESSOR
INSTRUCTIONS
270
254
MAIN MEMORY
INSTRUCTIONS
270
255
STATIC MEMORY
268
NETWORK INTERFACE DEVICE
NETWORK
BUS
256
258
VIDEO DISPLAY
260
ALPHA-NUMERIC INPUT DEVICE
262
CURSOR CONTROL DEVICE
264
DRIVE UNIT
265
MACHINE-READABLE MEDIUM
INSTRUCTIONS
270
266
SIGNAL GENERATION DEVICE

FIG. 12

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/AU2005/000454

This Annex lists the known "A" publication level patent family members relating to the patent documents cited in the above-mentioned international search report. The Australian Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent Document Cited in Search Report		Patent Family Member					
US	2004064447	DE	10328833	GB	2393541		
WO	0141002	AU	27253/01	AU	27254/01	US	6424358
		US	6523028	WO	0140919		
WO	0054185	AU	40070/00	EP	1212697		
WO	0065486	AU	63340/00				
US	2002169771	NONE					
WO	03030025	CA	2460717	EP	1430423	US	2004243595

Due to data integration issues this family listing may not include 10 digit Australian applications filed since May 2001.

END OF ANNEX